

AFRL-IF-RS-TR-2002-30
Final Technical Report
March 2002



PROACTIVE PROBLEM AVOIDANCE AND QUALITY OF SERVICE (QOS) GUARANTEES FOR LARGE HETEROGENEOUS NETWORKS

Lucent Technologies

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. F445

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.


The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2002-30 has been reviewed and is approved for publication.

APPROVED:



DANIEL J. HAGUE
Project Engineer

FOR THE DIRECTOR:



WARREN H. DEBANY, Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE			<i>Form Approved</i> OMB No. 074-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE MARCH 2002	3. REPORT TYPE AND DATES COVERED Final Sep 97 - Dec 99	
4. TITLE AND SUBTITLE PROACTIVE PROBLEM AVOIDANCE AND QUALITY OF SERVICE (QOS) GUARANTEES FOR LARGE HETEROGENEOUS NETWORKS			5. FUNDING NUMBERS C - F30602-97-C-0274 PE - 62301E PR - F445 TA - 00 WU - 01	
6. AUTHOR(S) William J. Hery				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Lucent Technologies, Incorporated 67 Whippany Road Room 15 B-243 Whippany New Jersey 07981-0903			8. PERFORMING ORGANIZATION REPORT NUMBER N/A	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency 3701 North Fairfax Drive Arlington Virginia 22203-1714			10. SPONSORING / MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-2002-30	
11. SUPPLEMENTARY NOTES Air Force Research Laboratory Project Engineer: Daniel J. Hague/IFGA/(315) 330-1885				
12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.				12b. DISTRIBUTION CODE
13. ABSTRACT (Maximum 200 Words) The two primary objectives of this research are: 1) To develop innovative, real-time "early warning" techniques to detect performance anomalies in large scale, heterogeneous networks that would not be detected by current single element network monitoring/alarm techniques, or to detect those anomalies before current techniques would detect them; and 2) To develop adaptive routing and traffic management methods which enable traffic to proactively avoid these potential problem areas before the problems become serious. This will not only improve the network performance seen by this traffic, but it will also prevent additional traffic from compounding whatever local problems were detected.				
14. SUBJECT TERMS Heterogeneous Networks, Performance Anomalies, Links, Nodes, Real-Time Early Warning Techniques				15. NUMBER OF PAGES 62
				16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

Table of Contents

Table of Contents.....	i
Summary	1
1. Introduction	2
1.1 Background/Significance	2
1.2 Architecture	3
1.3 Components	7
1.3.1 DOORS.....	8
1.3.2 Data Collection	8
1.3.3 Intelligent Agents	8
1.3.4 Visualization.....	9
1.3.5 Network Control.....	10
1.3.6 Design of Experiments (Wide Area Anomaly Detection).....	10
1.3.7 Proactive Connection Request Manager	11
1.4 Objective/Program Limitations.....	11
2. DOORS.....	13
2.1 Methods and Procedures	13
2.1.1 Nature/Procedure of Data Processed and Algorithms Used	13
2.1.2 Software Flow Description	13
2.1.3 Development Environment Description.....	14
2.2 Experiments and Results.....	14
2.2.1 Experimental Environment Description	14
2.2.2 Methods/Parameters Evaluated.....	14
2.2.3 Data Input/Output Description.....	15
2.3 Discussion	16
2.4 Conclusions and Recommendations	17
3. Data Collection.....	17
3.1 Methods and Procedures	17
3.1.1 Nature/Procedure of Data Processed and Algorithms Used	17
3.1.2 Software Flow Description	18
3.1.3 Development Environment Description.....	18
3.2 Experiments and Results.....	18
3.2.1 Experimental Environment Description	18
3.2.2 Methods/Parameters Evaluated.....	19
3.2.3 Data Input/Output Description.....	19
3.2.4 Summary of Results	19
3.3 Discussion	19
3.4 Conclusions and Recommendations	20
4. Intelligent Agents	20
4.1 Methods and Procedures	20
4.1.1 Nature/Procedure of Data Processed and Algorithms Used	20

4.1.2 Software Flow Description	23
4.1.3 Development Environment Description	23
4.2 Experiments and Results.....	23
4.2.1 Experimental Environment Description	24
4.2.2 Methods/Parameters Evaluated.....	24
4.2.3 Data Input/Output Description.....	24
4.2.4 Summary of Results	24
4.3 Discussion	25
4.4 Conclusions and Recommendations	26
5. Visualization.....	26
5.1 Methods and Procedures	26
5.1.1 Nature/Procedure of Data Processed and Algorithms Used	27
5.1.2 Software Flow Description	27
5.1.3 Development Environment Description.....	28
5.2 Experiments and Results.....	28
5.2.1 Experimental Environment Description	28
5.2.2 Methods/Parameters Evaluated.....	28
5.2.3 Data Input/Output Description.....	28
5.2.4 Summary of Results	28
5.3 Discussion	29
5.4 Conclusions and Recommendations	29
6. Network Control	29
6.1 Adaptive Routing	30
6.1.1 Methods and Procedures.....	30
6.1.2 Experiments and Results	34
6.1.3 Discussion	46
6.2 Traffic Management	46
6.2.1 Methods and Procedures.....	47
6.2.2 Experiments and Results	48
6.3 Conclusions and Recommendations	49
7. Design of Experiments.....	49
7.1 Methods and Procedures	50
7.1.1 Nature/Procedure of Data Processed and Algorithms Used	50
7.1.2 Software Flow Description	50
7.1.3 Development Environment Description.....	50
7.2 Experiments and Results.....	50
7.3 Discussion	51
7.4 Conclusions and Recommendations	51
8. Proactive Connection Request Manager	51
8.1 Methods and Procedures	51
8.1.1 Nature Procedure of Data Processed and Algorithms Used	52
8.1.2 Software Flow Description	52
8.1.3 Development Environment Description.....	52
8.2 Experiments and Results.....	53
8.2.1 Experimental Environment Description	53
8.2.2 Methods/Parameters Evaluated.....	53

8.2.3 Data Input/Output Description.....	53
8.2.4 Summary of Results	53
8.3 Discussion	53
8.4 Conclusions and Recommendations	53
9. Project Conclusions and Recommendations.....	54
References	54
Symbols, Abbreviations and Acronyms.....	55

Table of Figures

Figure 1.2.1 Top level Functional View	4
Figure 1.2.2 System Architecture.....	5
Figure 2.1.2.1 DOORS Process and Data Flow.....	14
Figure 2.2.2.1 DOORS Performance Experiment	15
Figure 2.2.4.1 DOORS Summary of Results	16
Figure 4.1.1 Agent Structure.....	21
Figure 6.1.1.1 Routing Control in Proactive Network Management Framework.....	31
Figure 6.1.1.2 Algorithm of Metricman	33
Figure 6.1.2.4.1 NSFNET backbone topology.....	36
Figure 6.1.2.4.2 Link Utilization, before and after activation of Metricman.....	37
Figure 6.1.2.4.3 Packet Loss, before and after activation of Metricman	37
Figure 6.1.2.4.4 Average UDP delay, before and after activation of Metricman	37
Figure 6.1.2.4.5 TCP round trip time, before and after activation of Metricman.....	37
Figure 6.1.2.4.6 TCP retransmission	38
Figure 6.1.2.4.7.....	39
HNcost and Metricman comparison - total packet loss	39
Figure 6.1.2.4.8 HNcost and Metricman comparison -end-to-end delay	39
Figure 6.1.2.4.9 Link cost evolution with HNcost	40
Figure 6.1.2.5.0 The N22_L30 topology. Link Utilization for N22_L30.	42
Figure 6.1.2.5.1The N26_L39 topology. LinkUtilization for N26_L39.	42
Figure 6.1.2.5.2 Metricman under different traffic	43
Figure 6.1.2.5.3 Metriman with different traffic localities	43
Figure 6.1.2.5.4 Metricman with long lasting TCP traffic.....	44
Figure 8.1. Software Flow Description	52

Table of Tables

Table 6.1 Link cost changes by Metricman for selected links	40
Table 6.2 Metricman parameter recommended range.	45

Preface

This report documents the work performed and results obtained under Contract Number: F30602-97-C-0274, "Proactive Problem Avoidance and QoS Guarantees for Large Heterogeneous Networks", sponsored by the Defense Advanced Research Projects Agency and administered by Rome Laboratory.

The contract was executed by Lucent Technologies Inc. Government Solutions (Lucent) of Whippany, New Jersey, with Rensselaer Polytechnic Institute (RPI) of Troy, NY, and Pennsylvania State University (PSU) of State College, PA, as subcontractors.

Deliverable software developed under this contract is documented in the Software Documentation Report (CDRL A005R).

Work on this contract was begun on September 15, 1997 and completed on December 10, 1999. This final report was completed in April 2000.

Summary

The primary goal of this program was to use real world data from the Lucent production Intranet to:

- Develop and evaluate novel approaches to detecting anomalies in large scale network, and
- Develop and evaluate network management techniques to maintain Quality of Service (QoS) in the presence of these anomalies.

The primary results of this program were:

1. A robust, distributed system to collect Management Information Base (MIB) data from network elements, such as routers.
2. Agents which analyze this MIB data to provide early detection of anomalies in a network. These agents detect anomalies, which are not detected by normal system alarms, and detect conditions, which may later lead to normal system alarms before those alarms are generated.
3. Adaptive routing techniques which improve the network performance seen by traffic in the presence of anomalies detected by the agents.
4. Adaptive flow control and admission control techniques which reduce load in network areas where anomalies have been discovered; this results in better overall network performance seen by the traffic.

Prototypes of the first two items (data collection and agents) were developed and deployed within the production Lucent Intranet. The third item, adaptive routing,

was evaluated by simulation, and the fourth item, adaptive flow control, was evaluated by analytic means.

1. Introduction

1.1 Background/Significance

The management of extremely large (>100,000 node) heterogeneous networks requires a novel approach—one that moves away from the current strict hierarchical manager of managers approach based on cooperative networks. In fact, network management today is, in reality, more monitoring than management, with diagnosis and solution often provided by a human network manager. In this program, we developed an approach for creating an automated, distributed network management system of novel network problem detection, diagnosis, and management components. These work together to (1) provide pro-active detection of network problems, (2) assist in the avoidance of these problem areas by network traffic. A third aspect, to provide a mechanism to ensure rapid satisfaction of connection requests with quality-of-service requirements, was part of the first year of the program, but was not continued.

The two primary objectives of this research are:

1. To develop innovative, real-time “early warning” techniques to detect performance anomalies in large scale, heterogeneous networks that would not be detected by current single element network monitoring/alarm techniques, or to detect those anomalies before current techniques would detect them; and
2. To develop adaptive routing and traffic management methods which enable traffic to proactively avoid these potential problem areas before the problems become serious. This will not only improve the network performance seen by this traffic, but will also prevent additional traffic from compounding whatever local problems were detected.

The performance anomalies to be detected include link, node, and server failures; local congestion due to new services, “special events” (such as the release of popular new software on a server); and certain types of security events, such as localized denial of service attacks.

Realism of the methods and models developed is greatly enhanced by the extensive use of data from Lucent’s large scale, production Intranet as the basis for developing the anomaly detection techniques, and meaningful simulations to assess the applicability of those techniques.

To support this research and as a basis for deployment of future tools, a distributed object framework (DOORS) was developed to provide a robust data collection and warehousing facility to support the analysis tools. A simple network visualization tool was also developed.

The techniques explored in this program were explored and evaluated via either simulation or the development of test code for small scale experimental deployment. However, each technique was explored with the eventual goal (outside the scope of this program) of developing software implementations which could be widely deployed. Section 1.2 describes the system architecture of an integrated set of these future deployable components. An understanding of this long term future architecture is key to understanding the work done for each of the individual components.

Section 1.3 describes the major components of this program, and identifies the organization with primary responsibility for each component.

1.2 Architecture

A top level functional view of the major elements of this program is shown in Figure 1.2.1. Two anomaly detection components are shown on the left:

- Local anomaly detection, performed through the use of Intelligent Agents (IA's) processing local information at network elements, such as routers, switches, gateways, etc. which are observable to our system.
- Wide area anomaly detection, performed through the use of network probes through unobservable regions of a network.

Three distributed management components are shown on the right:

- Proactive problem avoidance methods, which use network control (routing algorithms, flow control, admission control) to reduce the traffic in problem areas.
- Display and visualization, which allows network administrators to monitor the state of the network.
- Proactive connection requests, which use network status information to select the best current connection to supply a desired service.

The distributed objects framework is shown in the middle, but this is an oversimplification: it is truly distributed throughout the network, and (for example) the IA's use the distributed object framework to request local network data.

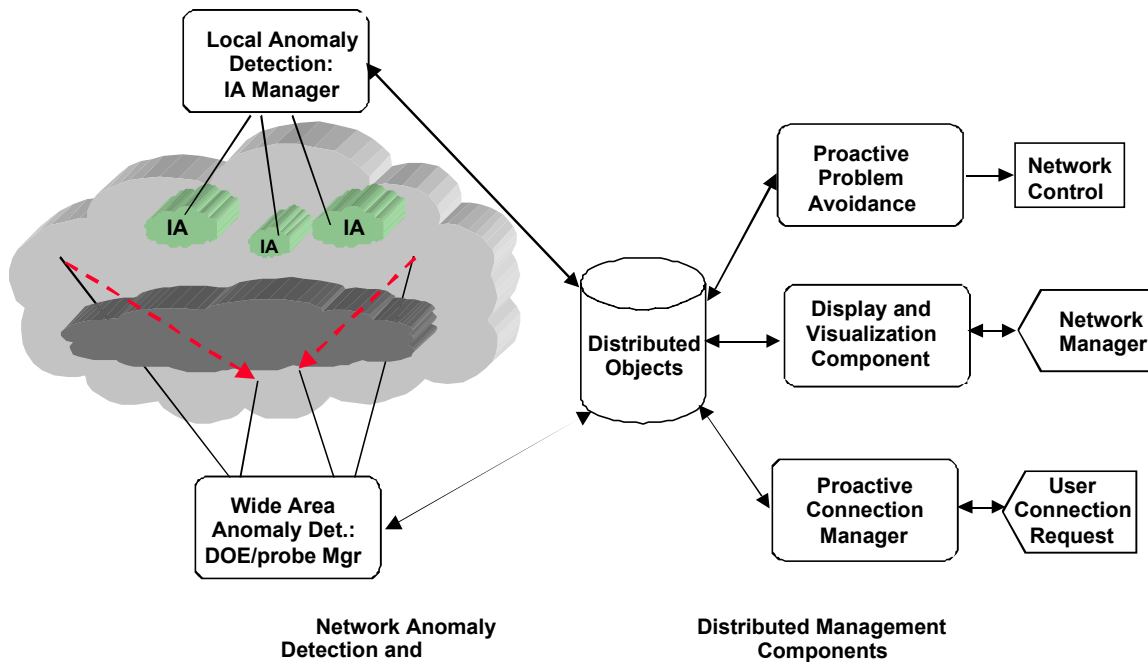


Figure 1.2.1 Top level Functional View

The elements shown in Figure 1.2.1 were developed to varying degrees under this program. For some, such as the Intelligent Agents for local area anomaly detection, and DOORS, working prototype software was developed and delivered to DARPA. Others, such as the network control elements, were designed and evaluated only through simulations. The long term goal, of course, is that all components be developed and deployed in real networks.

A more detailed architectural view of deployable versions of these components and their communications paths in a network is shown in Figure 1.2.2. The operation of these components will be described in this report.

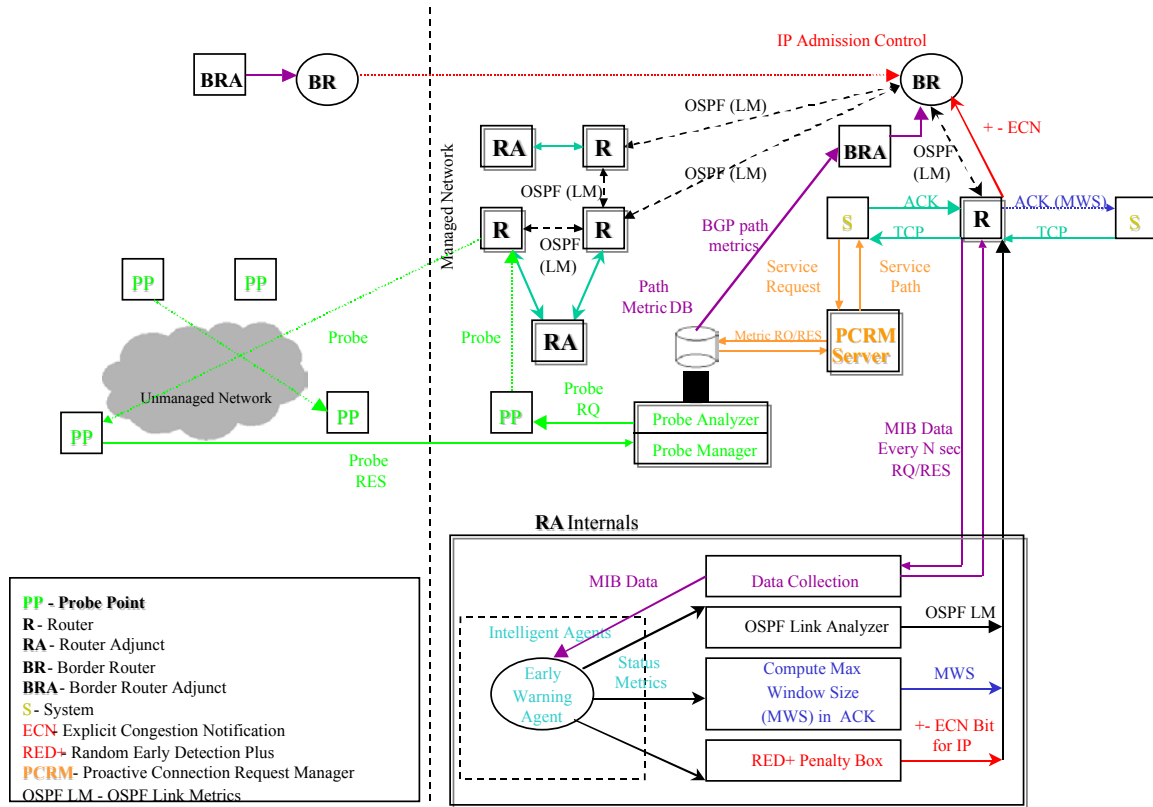


Figure 1.2.2 System Architecture

Figure 1.2.2 shows the location of the various components in a fully deployed configuration on a large network. Portions of the network not directly observable are shown in the cloud on the left, along with a set of probe points (“PP”) where probes are sent through the network for collecting network performance information. Observable portions are shown in greater detail on the right, with the observation and response elements shown. Typically, the observable portions are those under the administrative control of a single organization, such as corporate intranet, or an ISP backbone. The unobservable portions are then “the rest of the world of interest”. We further assume (for simplicity) that the observable portions are the same as the controllable portions of the network.

The heart of the deployed system for the observable portion is the set of router adjuncts (“RA”), which

- Perform the collection of data from the observable network elements using DOORS,
- Perform the analysis of that data using the Intelligent Agents (IA’s),
- Provide the information for the Network Control response techniques, and
- Provide the status information for visualization and network management.

A single RA runs on any networked host, and can be used to monitor and provide problem response data to multiple network elements.

A blowup of the components of an RA is shown in the lower right portion of the figure. The operation of an RA is as follows: The IA posts a request to DOORS to collect data; the request includes the elements to collect from, the specific MIB variables to collect, and the frequency of collection. DOORS then uses this information to begin polling the network elements, and sending the data to the IA as it is collected. The IA processes the data as it is collected, and after each processing window for each element it generates a set of status values for that element. That status value is then sent to various network control and network management processes for application.

Network Control via adaptive routing is based on the adaptation of the Open Shortest Path First (OSPF) routing protocol by incorporating information from the IA into the OSPF link metric. For the OSPF link metric, the IA status values are used to construct a set of OSPF link metrics for each link at the network element. That data is then sent to other routers in the same autonomous system to be used by OSPF to compute up to date routing tables reflecting the current status of network elements.

For Network Control via adaptive flow control, the IA status value is used to compute the maximum window (MWS) size for TCP connections flowing through the network element. The MWS is then sent to the network element (router) for application. This technique cannot be used on all routers, but requires the router be capable of processing TCP ACK packets traversing the router. On such routers, the MWS in each ACK packet is examined, and if it is greater than the computed MWS, the computed value replaces the old value. This will then throttle back the TCP traffic through that router, although it will not affect any UDP traffic.

For Network Control via admission control, the IA status value is used to compute a congestion parameter for that part of the network. That parameter is then sent to the boundary routers of the controllable portion of the network, which uses the set of congestion parameters within the network to determine admission of external traffic to the network.

Last, the status information is sent to the traditional network management tools, such as the visualization tool prototyped here.

For the monitoring and optimized use of the unobservable network regions, probes of that region are used. These probes are launched from a suite of probe points ("PP"), and the results collected at the same or other PP's. These PP's may be within the controllable network region, or they may be located at cooperating sites.

To manage this, there is a probe manager within the controllable portion of the network. The probe manager maintains information about the location of the PP's, dispatches probe requests (which may be for periodic probes, or special localization probes), and collects the probe results.

The probe results are then passed to the probe analyzer, which performs the following functions:

- Use DOE selection techniques to determine which probes to execute for optimal information collection.
- Use DOE correlation techniques to build a view of the entire network.
- Use DOE time of day models to build a projected view of the network.
- Populate a path metric database (PMDB).

The PMDB is then available as a resource for the Proactive Connection Request Manager (PCRM) Server. When a Requesting System within the network requests a service registered with the PCRM Server, the server is contacted to determine the preferred connection to make by the Requesting System to the requested service. The PCRM Server uses the performance metrics in the path metric database to select the preferred connection based on performance requirements, current network status, and potentially time of day and cost. The preferred path is then used by the Requesting System to connect to the service.

1.3 Components

This section describes the major components of this program:

1. DOORS, the distributed object oriented repository system that ties the components together and to the network.
2. Data collection from the Lucent Intranet, to provide a realistic basis for evaluating the techniques.
3. Intelligent Agents (IA's) for local anomaly detection.
4. Visualization of the IA output to monitor network conditions.
5. Network Control, to adapt network routing and traffic to anomalous situations.
6. Design of Experiments (DOE) for wide area anomaly detection by analyzing the performance of network probes.
7. Proactive Connection Request Manager, to select the preferred path to a resource given current network conditions.

This section will also identify the organization performing the work and the degree of development.

1.3.1 DOORS

In support of decentralized and proactive network management we have developed a Distributed Object-Oriented Repository System (DOORS). At the core, DOORS provides a secure, efficient, and fault tolerant method for the acquisition, management, manipulation, aggregation, and caching of network data and objects. We use mobile agents and a dynamic interface to support SNMP protocol.

DOORS is composed of a set of repositories responsible for managing data collection agents for a region of the network. The repositories sends agents to polling station which usually are set up close to the managed devices to securely gather data. The polling stations allow agents to poll from a very short network distance no matter where the client or visualization components may be. This insures reliable and efficient data gathering. The repositories combine and then schedule execution of requests to reduce the amount of monitoring traffic that the data collection imposes on the network.

DOORS collects the data on behalf of the Intelligent Agent (IA) and Visualization components of the system.

DOORS was developed by RPI, and prototype code with partial functionality was completed. This work was only partly funded under this program; primary corporate funding was provided by IBM under a separate contract.

1.3.2 Data Collection

Prior studies of network anomaly detection have been based primarily on theoretical expectations of behavior in the presence of anomalies, or on data collected from lab networks. This program, however, is based on data collected from the Lucent production intranet over an extended period of time. In particular, local anomaly detection (IA) development is based on:

- Management Information Base (MIB) data collected from a production router in the Lucent Intranet,
- Syslog information collected from servers on the subnets being monitored by the IA's, and
- Trouble ticket reports sent by the users of the subnets being monitored.

Data collection was performed by Lucent, and the data used in the development of the IA component.

1.3.3 Intelligent Agents

The role of intelligent agents we developed in this work is to generate status values (alarms) to indicate potential network anomalies by (automatically) processing aggregated traffic measurements available at network nodes. An intelligent agent gets measurements from a network node (provided by DOOR),

Generates an alarm if an anomaly is present, and passes the alarm to network control components.

The core of the intelligent agent is an algorithm for processing measurements to generate an alarm. The algorithm has been developed based on statistical signal processing and learning. There are two salient features on the intelligent agents developed: (1) an agent detected network anomalies, which manifest themselves in network traffic, and (2) the detection was done on-line (in real time). Due to these properties, intelligent agents have shown to be able to perform proactive detection on network anomalies before they cause catastrophic network problems.

The research on developing the intelligent agents also provides insights and understanding on (a) what anomalies can be detected by intelligent agents, and (b) when and whether it is possible to detect network anomalies early (proactively).

The IA component was developed by RPI. Operational prototype code was developed.

1.3.4 Visualization

The visualization tool provides a graphical display of the IA view of the health of the routers and interfaces being monitored. The health of the router and interfaces are represented as a gradient of colors to provide a quick view of which areas on the network problems may be forming. For multiple routers, the physical connections between interfaces along with the respective health of each interface are represented.

In addition to viewing the current status, the user may also record network information as it is being displayed. This recorded information can be accessed by the playback module, which allows the user to view any of the recorded information using a sliding menu bar.

The code is a series of script files written in Matlab that allows the visualization tool to be platform independent. In addition, it allows for quick proof of concept prototyping of new features that can eventually be added to a standalone executable.

This tool was developed for demonstration and testing purposes on a small network only. In a deployed network, the data would be fed into the overall network management package (such as HP OpenView) being used. In that way the sophisticated network capture and representation software of package can be utilized, and the IA information can be integrated with other, standard network information.

The visualization tool was developed by Lucent Technologies. Prototype code was developed.

1.3.5 Network Control

Closely linked to the IA local anomaly detection are our adaptive routing, flow control, and traffic management techniques for proactive problem avoidance. Adaptive routing is a proactive approach to problem avoidance by routing packets away from “problem areas”. In our research, problem areas are indicated by the IA status values; these values are available at routers for this purpose. OSPF is a widely used method for building routing tables based on standard link metrics. We developed new link state metrics based on the IA outputs to be used by OSPF; this enables traffic to be automatically routed around impending problems before they become serious.

Traffic management controls the amount of traffic in a “problem area” by cutting back the traffic, rather than re-routing it. Two methods of traffic management were examined. Adaptive flow control for TCP/IP sessions is achieved by using the IA output to determine a “fair share” maximum window size for traffic through the router the IA is monitoring; that size will replace larger window sizes in TCP ACK packets passing through the router. Finally, admission control to a network region is limited by ingress routers in the presence of congestion: IA output indicating a congestion threshold has been exceeded results in setting the single bit Explicit Congestion Notification (ECN) on IP headers. This information is then sent to the ingress routers, where admission to the network is limited in response to the congestion.

The network control components were developed by RPI. Each of the components was evaluated mathematically and/or via simulation.

1.3.6 Design of Experiments (Wide Area Anomaly Detection)

Wide area anomaly detection and network characterization is based on automated network performance probes sent at frequent intervals between widely distributed probe points (PP's). These probes may go through unobservable, unmapped regions of the network, although the PP's themselves must be known and accessible. To provide a scalable approach to network probing, mathematical and multivariate statistical techniques such as Design of Experiments (DOE) and topological aggregation are used to minimize the number of source/destination probe pairs needed. Data collected over time is used to develop and maintain a view of the “normal” behavior of the network as a function of TOD/DOW. Frequent probes are then used to identify any pattern of significant deviation from “normal” performance as an indication of a network anomaly. Normal and current performance characteristics are maintained in a distributed WAN performance database which is publicly available. These probes and the correlation functions are also a first step toward developing a network tomography.

In the early course of this work, it was found that the existing background research on IP network modeling, traffic distributions, and correlation analysis of performance of adjacent network elements (such as routers) did not provide an adequate basis for the planned DOE approach. The task was re-defined to include this research and additional modeling and correlation studies were begun. One year into the program, however, it was decided that there was not sufficient funding to make adequate progress on both this modeling and the original DOE task, and work in this area was stopped at the request of DARPA.

Work on just the mathematical approaches to select optimal sets of probes was continued by PSU under university funding after DARPA funding ceased. This report documents the work performed under DARPA funding. Part of this work and additional university funded research is described in [DOE1].

The DOE component was developed by PSU.

1.3.7 Proactive Connection Request Manager

The performance characterizations developed in the DOE (wide area anomaly detection) form the basis for another problem avoidance technique: the Proactive Connection Request Manager (PCRM). The PCRM is used to select ingress (and possibly egress) points for a network to network connection through a large WAN without using any direct observations of the WAN interior. For example, a large corporate network may wish to establish a connection with another corporate network through the Internet. Each corporate network may have several connections to the Internet, including connections to different ISPs and high cost, usage priced connections. The PCRM responds to connection requests from the source network by using both the TOD/DOW and current performance information in the WAN performance database to select the lowest cost connection meeting the connection Quality of Service (QoS) requirements.

The primary basis for evaluating the QoS of alternate paths considered by the PCRM is the performance characteristics supplied by the DOE models. Shortly after the DOE component was stopped, it was decided that the PCRM should also be stopped, and the funding planned for this task was used to develop a visualization tool for the IA's.

The PCRM work was performed by Lucent Technologies.

1.4 Objective/Program Limitations

The objectives of this program were to develop and evaluate anomaly detection and response techniques, and in particular to evaluate these techniques working together, as described in this document. One limitation of this research is that deployable code was not developed: only the DOOR and IA components were

implemented in a form that has actually been run on real networks. Other elements have been evaluated by simulation and analysis.

These techniques, however, were designed to be techniques which *could* be implemented and deployed on the Internet and intranets. In practical terms, there are institutional and practical limitations on such deployments:

Intelligent Agents for local anomaly detection depend on access to router level MIB information and local network topology. Such access would be limited to those with administrative control of the router. In addition, proper use of the IA information would be a part of the network management of the router. Therefore, deployment of IA's would be practical in corporate, government or university intranet works with centralized administration, or in portions of the public Internet under a single administration, such as an ISP or a backbone provider. Ubiquitous deployment in the Internet is unlikely.

Adaptive Routing for network control depends on the adaptation of routers to use these techniques. The OSPF technique evaluated in this research depends on modification of the router software to accept non-standard link metrics for OSPF, as well as the administrative control to turn on such a feature and accept values from an IA monitoring that router. Since OSPF is designed to run within an Autonomous System (AS), and AS's are typically under a single administrative control, the primary issue is the availability of routers with software adapted to accept non-standard link metrics.

Traffic Management for network control also depends on modification of router or border router software as well as administrative control to turn on such features.

DOE for wide area anomaly detection is designed to avoid the problems of local control by sending probe packets through unobservable portions of the Internet. This requires two elements: a set of distributed "probe points" that can be used to launch network probes and a probe that routers in the will accept as valid traffic to be routed with other traffic. Since a relatively small number of probe points are needed, it is reasonable to postulate that a consortium of interested parties (ISPs, universities, corporate and government sites) could be built to support such an effort when its value was proven. Smaller scale projects, such as that being run by CAIDA, are already doing that to some extent. Probes may become problematic: as security concerns grow, more and more routers are being configured to screen and drop packets which may have negative security implications. In particular, two common probes (ping and traceroute) are sometimes used to probe networks in search of security holes, and are sometimes blocked.

2. DOORS

2.1 Methods and Procedures

2.1.1 Nature/Procedure of Data Processed and Algorithms Used

DOORS retrieves information from managed devices using a distributed mobile agent approach. When a request is made from the client, the Repository checks to see if an agent is already collecting the necessary data. If this is the case, the repository simply sends a copy of this data to the new requesting client.

However, if no agent is collecting the necessary data, an agent is created and dispatched to do so. This new agent is directed to go to the polling station and poll for the desired information. The results of the poll are sent back to the repository who sends the results to the requesting clients. Once the desired duration of data collection has expired, the agent returns to the repository where it is disposed. The flow of data can be seen in the illustration in the following subsection.

2.1.2 Software Flow Description

The procedure and flow of data discussed in the previous subsection can be visualized in Figure 2.1.2.1. Note that one polling station can be responsible for more than one router. Also note that the repository can be responsible for more than one client. Not pictured here is the situation where the client wants information from a router outside the repository's domain. In this case, the local repository would act as a client for a repository whose domain includes the router in question. The information will in turn be passed on to the actual client via the

local repository.

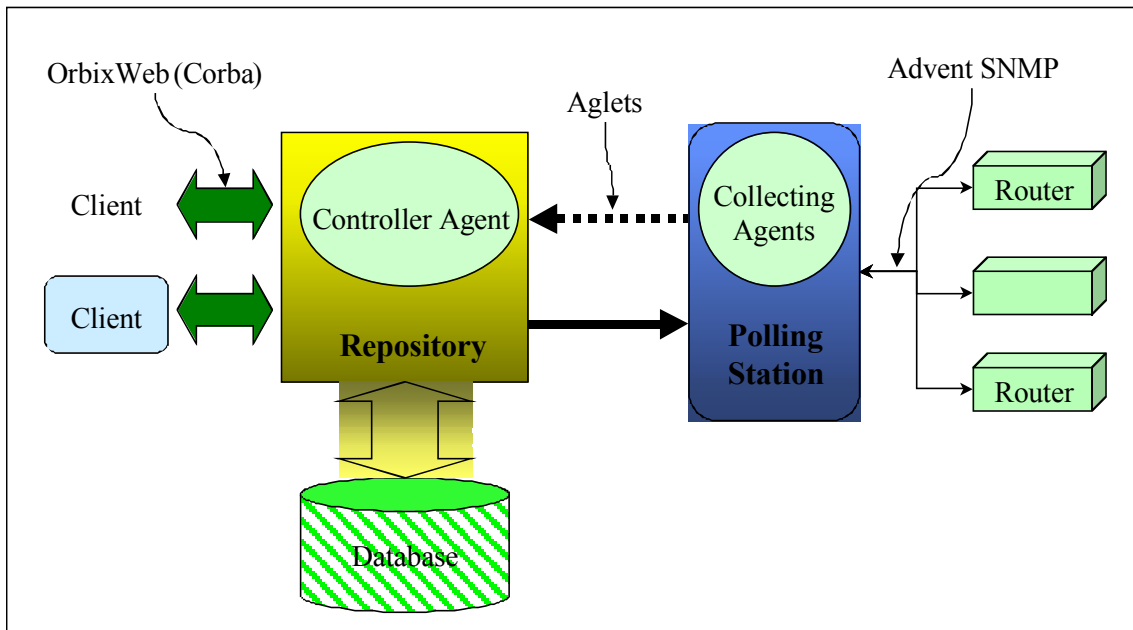


Figure 2.1.2.1 DOORS Process and Data Flow

2.1.3 Development Environment Description

DOORS was created in a computer laboratory of SUN workstations. Most of the development was done on 2 Sparc Ultra2's, and several Sparc 20's. The development environment was the same as the initial testing environment. It is pictured in the next section.

2.2 Experiments and Results

2.2.1 Experimental Environment Description

All initial experiments were conducted at the Computer Science department at Rensselaer Polytechnic Institute. The experiments were performed to determine the extent of overhead that the system imposes on the routers and the network. In the first set of measurements, we used a simple SNMP procedure to poll from a long distance (the way many applications are written to do). The second set of measurements used Aglets agent package to send an agent to a polling station to poll a router. The last set of measurements was using both CORBA and Aglets (i.e., the DOORS system). There were up to four machines involved in the experiments. All of the machines were on the same subnet (cs.rpi.edu) except *paris*, which was 2 hops away (it belongs to another subnet, it.rpi.edu). An illustration of the experiment configurations is given below, followed by plots of the results.

2.2.2 Methods/Parameters Evaluated

To emulate the direct way of polling in DOORS for a good comparison with direct and agent based data collection, we formed the DOORS clients to poll the

repository as it was populated with information. The timing was compared between the different cases shown in Figure 2.2.2.1.

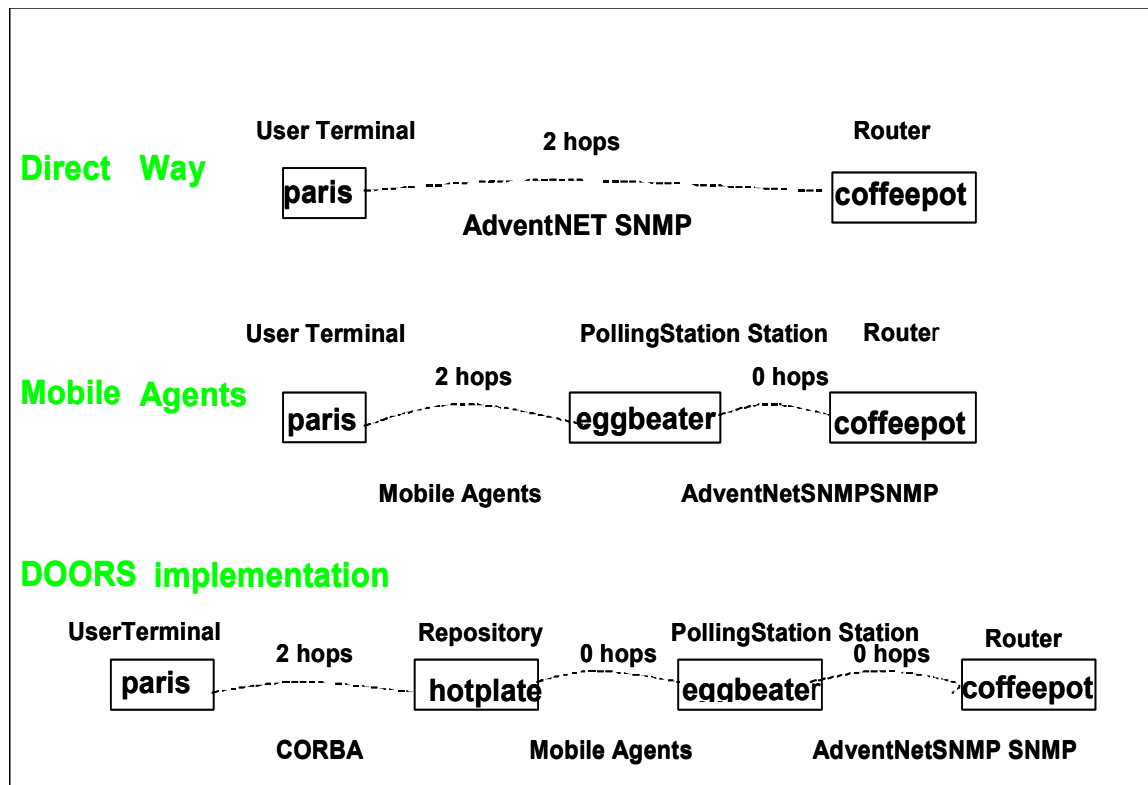


Figure 2.2.2.1 DOORS Performance Experiment

2.2.3 Data Input/Output Description

The input data was simply the configuration file of the client describing the information to be collected. The output was the collected SNMP data with timestamps of their collection.

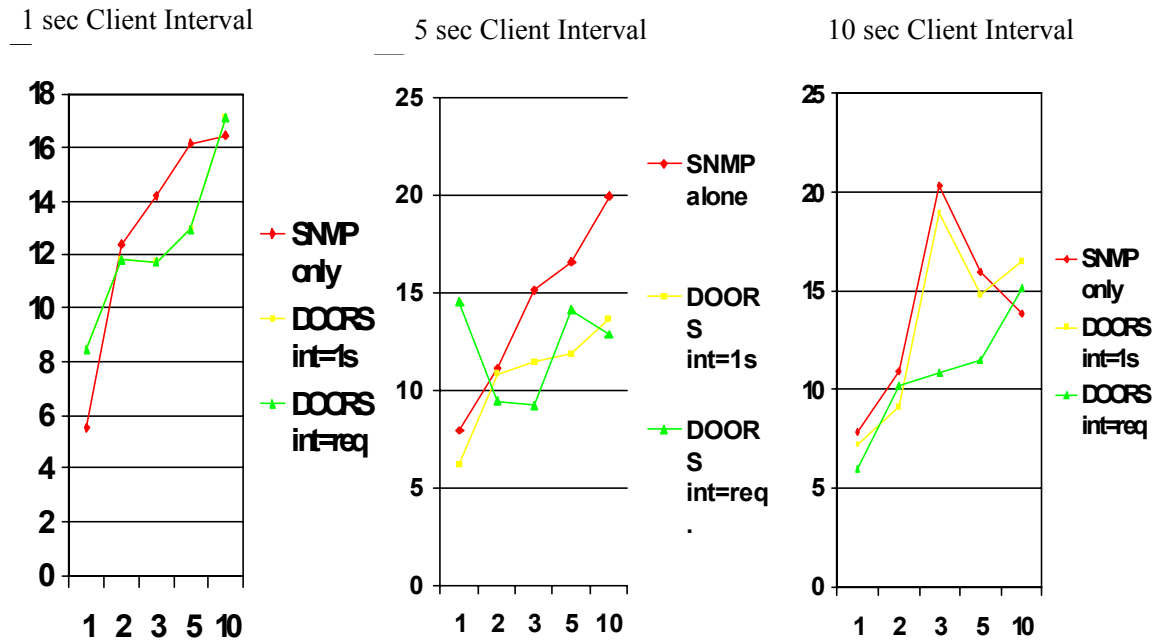


Figure 2.2.4.1 DOORS Summary of Results

Figure 2.2.4.1 provides a graphical summary of the results. The results are different, depending on the polling rate of the clients. The x-axis defines the number of clients requesting data. The y-axis represents time. The line with the squares in it is the DOORS system running with a polling rate of 1 second. The triangle line represents time when the client polling rate is the same as the polling rate set by the repository.

2.3 Discussion

In cases shown, DOORS either outperforms or performs at comparable level as the direct SNMP query method. Sometimes the direct SNMP query method fails to return a value because of heavy polling from several clients. In the DOORS system, this happened much less frequently because only one agent is used to query the router.

In these tests, DOORS, was configured to simulate the direct SNMP method of retrieving data. When the data is pushed to the client instead of the client polling it, even better results are expected. Our DOORS system was a bit of overkill for the closed local network testing that we were able to perform at the Computer Science Department. DOORS are more useful when deployed on a very large network. In very large networks, it is not feasible to use the direct SNMP method for monitoring because of the distance traveled by the request and responses. For such networks, a highly distributed architecture for collecting data, supported by the DOORS is necessary for efficient and non-intrusive data collection. The details of the experiments are presented in [DOOR1].

2.4 Conclusions and Recommendations

Our conclusion is that DOORS is a tool, which is useful for collecting vast amount of data over large computer networks, when applications need to overstep the limiting bounds of the normal methods of data collection. An important DOORS ability is to perform simple calculations or data transformation at the polling stations and repositories, so only processed data (potentially much of much smaller volume than raw data) need to be transmitted. In addition to collecting data, the DOORS system can also be used to distribution of alarms and signals for the network management centers to routers.

Another possible extension is to built a simple database in the repository to improve the storage and retrieval of historical data. We report the initial implementation of such a database system in [DOOR2].

3. Data Collection

3.1 Methods and Procedures

3.1.1 Nature/Procedure of Data Processed and Algorithms Used

Three types of data were collected to support the IA development:

- Management Information Base (MIB) data collected from a production router in the Lucent Intranet,
- SYSLOG information collected from servers on the subnets being monitored by the IA's, and
- Trouble ticket reports sent by the users of the subnets being monitored.

The *MIB data* is collected by the production router in the normal course of operation. To access that data, our polling station uses Simple Network Management Protocol (SNMP), and polls the router at regular intervals for the required data.

Initially, the router was polled every second, but research on the IA indicated that data at 15 second intervals was all that was needed. In phase two of the data collection, therefore, the polling frequency was reduced to 15 seconds.

Initially, approximately 50 MIB variables were collected from the router. The research on the IA indicated that only the following MIB variables were significant for the IA research and in phase two of the data collection only those variables were collected.

- At the router level (ip variables):
 - ipInReceives
 - ipInDelivers

- ipOutRequests
- For each interface (if variables):
 - ifTable.ifEntry.ifInOctets
 - ifTable.ifEntry.ifOutOctets

Initially, only data from two of the 36 interfaces on the router was collected. In phase two of the data collection, data from all 36 interfaces was collected.

The *SYSLOG data* is generated on each server automatically as a part of normal operation. To access the data, shell scripts were installed on the server to email the SYSLOG files to the polling station on a regular basis.

The *Trouble ticket reports* are a normal part of network management for the Lucent Intranet. Periodically, the incremental file of closed trouble tickets was emailed to the polling station.

Every week, the data collected was written to CD-ROM and sent to RPI for the IA development.

3.1.2 Software Flow Description

The MIB data was collected from the router by a remote workstation running a simple program to poll the router for the desired variables at the desired frequency; the data was then written to a file on disk. On a weekly basis, the data files were compressed, written to a CD, and shipped to RPI for analysis. In a production system, MIB data would be collected using the DOOR software, which is described in section 2.

The SYSLOG and Trouble Ticket information are collected via standard email, and no special software was developed. The files were also included on the CD sent to RPI.

3.1.3 Development Environment Description

Not Applicable.

3.2 Experiments and Results

3.2.1 Experimental Environment Description

Other than testing the DOOR software, the only experiment conducted in support of the data collection was an evaluation of the performance impact on router. This was necessary because the data was to be collected from a production router, and performance impacts would not be tolerated by the users of that router.

There were two potential performance issues: traffic load and router CPU loading. Traffic loading was computed directly, and no experiments performed. An experiment was designed to measure CPU utilization for the data collection task. Experiments were performed initially on a Sun workstation acting as a router at RPI, replicated on a similar workstation at Lucent, and then run on the Lucent production router to demonstrate that the performance impact was negligible.

3.2.2 Methods/Parameters Evaluated

The parameter evaluated was a router CPU utilization for data collection. The data collection software was run at a one second polling rate, and the router CPU utilization attributable to that task was measured.

3.2.3 Data Input/Output Description

The input to the polling program was the set of MIB variables to collect, and the polling frequency. The output was the router CPU utilization attributable to the polling process.

3.2.4 Summary of Results

The utilization was shown to be less than one tenth of one percent in each environment. Since this was negligible, no further experiments were conducted.

3.3 Discussion

The greatest difficulties in collecting the data were non-technical. The cooperation of three different organizations at Lucent (in addition to the organization executing this program) was needed to collect the data:

- The Lucent CIO organization has responsibility for the Lucent production routers. After explaining the potential value of the research and the negligible performance impact, permission was granted to collect data from one production router.
- Servers and desktop computers are managed by ISSC (a part of IBM) under an out-sourcing contract from Lucent. They agreed to set up a process to automatically collect and forward SYSLOG messages from servers on the subnets being monitored. They also agreed to manually forward batches of closed trouble tickets.
- The Lucent Wireless R&D organization is the primary user of the subnets being monitored. They also were presented with the advantages and performance implications of the data collection and agreed to permit it.

The data in its raw form is Lucent Proprietary, and is not a deliverable under this contract. Only conclusions drawn from the data by analysis, presented in a form that masks Lucent's proprietary and confidential information are included in this report.

3.4 Conclusions and Recommendations

The data collected is a valuable resource for developing models of network behavior and techniques for anomaly detection. The value of this data comes not only from its usefulness in analysis, but from its rarity: most organizations would not allow this degree of data collection from production systems for purely research purposes. It is recommended that ways be sought to make use of this information in other research projects, although the proprietary nature of it means the data cannot be distributed.

4. Intelligent Agents

The role of intelligent agents we developed in this work is to generate status values (alarms) to indicate potential network anomalies by (automatically) processing aggregated traffic measurements available at network nodes. An intelligent agent gets measurements from a network node (provided by DOOR), generates an alarm if an anomaly is present, and passes the alarm to network control components.

The core of the intelligent agent is an algorithm for processing measurements to generate an alarm. The algorithm has been developed based on statistical signal processing and learning. There are two salient features on the intelligent agents developed: (1) an agent detected network anomalies, which manifest themselves in network traffic, and (2) the detection was done on-line (in real time). Due to these properties, intelligent agents have shown to be able to perform proactive detection on network anomalies before they cause catastrophic network problems.

4.1 Methods and Procedures

4.1.1 Nature/Procedure of Data Processed and Algorithms Used

The IA performs the following steps:

- Receives a time sequences block of value for each MIB variable.
- Performs Auto-regressive (AR) time series analysis on each block of MIB data.
- Derives a “change detector” value for each MIB variable.
- Combines the change detector values for all the MIB variables to get an overall element status value.

Figure 4.1.1 below illustrates this process.

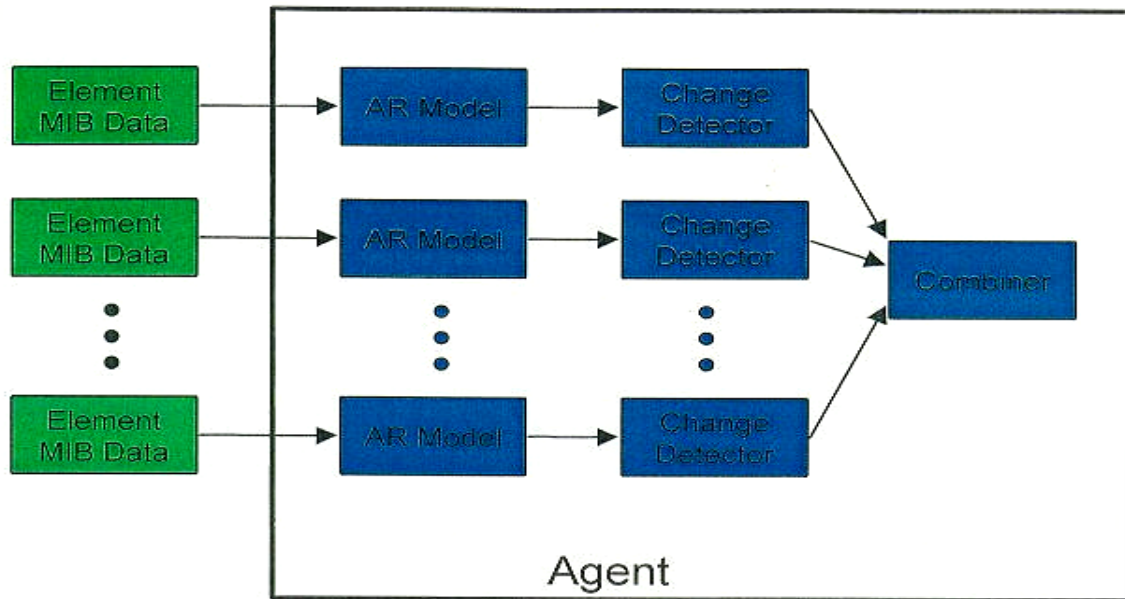


Figure 4.1.1 Agent Structure

4.1.1.1 Choice of MIB Variables

The traffic-related measurements are collected through SNMP agents. The SNMP protocol maintains a set of counters known as the Management Information Base (MIB) variables. The Management Information Base maintains 171 variables for one of the routers from which we collected data. These variables fall into different groups, some of which are not traffic related. The choice of an optimal set of MIB variables that are relevant to the detection of traffic related problems helps reduce the computational complexity by reducing the dimensionality of the problem.

A subset of these variables were chosen based on their ability to capture the traffic flow into and out of the device. In this work the node being monitored was a router, and therefore if and the ip group of variables were investigated. The if groups of variables describe the traffic characteristics at a particular interface of the router and the ip variables describe the traffic characteristics at the network layer. By removing redundant MIB ip and if variables, we selected five MIB variables for our work: iflO (In Octets), input to an interface; ifOO (Out Octets), outgoing traffic from an interface; ipIR (In Receives), the total number of datagrams received from all interfaces of a router; ipIDe (In Delivers), the number of datagrams correctly delivered to the higher layers as this node was their final destination ipOR (Out Requests), the number of datagrams passed on from the higher layers of the node to be forwarded by the ip layer.

The ip variables sufficiently describe the functionality of the router. The if layer variables help to isolate the problem to the finer granularity of the subnet level. Details on choices of MIB variables can be obtained from [IA1].

4.1.1.2 Change-Detector and Combiner

Intelligent agents are software entities that process the raw MIB data obtained from the devices to provide a real time indicator of network health.

Two types of agents were developed: Interface agents which process if variables, and router agents which process ip variables. The interface layer processing or the if-agent yields the performance indicator of a specific subnet connected to a particular interface of the router while the ip-agent provides the performance indicator at the network level. As their development is similar, we focus on describing ip agents.

An ip-agent consists of a change-detector and a combiner. We assume that network anomalies manifest themselves as abrupt changes in the traffic related MIB variables. The abrupt changes associated with these anomalies can then be detected as changes in the behavior of the time series data obtained from the MIB variables. Since the MIB variables have different statistical properties, for each individual MIB variable a sensor (change-detector) was designed to perform the required statistical signal processing.

Considering one sensor (change-detector). The time series data obtained from the MIB variables were non-stationary; hence we required an adaptive learning algorithm to account for the normal drifts in the traffic. Measurements from two adjacent non-overlapping windows of the time series were considered, a learning window $L(t)$ and the test window $S(t)$. The length of these windows is chosen so that the time series data within these windows could be considered piece-wise stationary. As time increments, these windows slide across the time series. A model for each stationary piece was assumed to be to be an Auto-Regressive (AR) process. AR parameters were extracted using measurements from each window. A sequential hypothesis test was performed on the residual errors to determine whether a change had occurred going from the learning window to the test window. The details on hypothesis testing can be obtained from [IA1]-[IA4] and the references therein.

The output of each sensor (change-detector) was the likelihood of abnormality for each of the MIB variables. These indicators, which were functions of system time, are updated every $N(S)$ lags.

Since the MIB variables are not strictly independent, they have cross correlations. As the sensor outputs were obtained by treating the MIB variables independently, the outputs of the sensors needed to be combined to take into account these dependencies. In other words, we need to develop a method for identifying correlated changes in the MIB variables when network anomalies

occur. This task was accomplished using a fusion center. The fusion center was used to incorporate these spatial dependencies into the time correlated variable level abnormality indicators.

The design of the fusion center is based on correlation among chosen MIB variables. In particular, a matrix, as a linear operator, was designed for IP variables as an example, where elements of the matrix were normalized sample correlation of the three chosen MIB variables of the ip group. The main diagonal terms are normalized such that the rows and columns sum to 1. The values of the elements were calculated using data from the campus network. (The values used for the enterprise network were the same as those of the campus network.)

We used the eigen-vectors of the matrix to define the faulty region of the space, and the eigen-values as thresholds to declare alarms. Details can be found in [IA5]. The output of the fusion center was then a single continuous scalar indicator of network level abnormality as perceived by the node level agent through a linear operator.

4.1.2 Software Flow Description

The software, which implements the agent algorithm, consists of two parts:

- A change detection algorithm for each MIB variable, and
- A fusion center which combines the outputs from all MIB variables.

An off-line and an on-line agents have been implemented. Basic ideas for both versions of agents were the same except the former processes measurements stored in a data file, and the latter generates alarms on-line when measurements are being collected a small patch at a time.

A flow chart is included in the software descriptions as an illustration of on-line agents (the idea for off-line agents is similar, and the corresponding source code (in matlab) can be found in [IA5]).

4.1.3 Development Environment Description

The off-line version of the agent algorithm was written in matlab. The program can run at any environment with matlab. The on-line version used a C-wrapper on the matlab in order to run multiple copies of intelligent agents. Details can be obtained from software descriptions.

4.2 Experiments and Results

4.2.1 Experimental Environment Description

Data collected from two networks, an enterprise network and a campus network, were used to test (the off-line version of) the agent. The topology of the enterprise network and that of the campus network can be found in [IA1]-[IA4]. The corporate network was significantly larger than the campus network.

Measurements on the five MIB variables were collected from a router, and one Interface at the router at a sampling rate of every 15-second interval. They were then processed using the off-line agent.

4.2.2 Methods/Parameters Evaluated

Algorithms for change detection and fusion center were tested using the data. The goal for the experimental evaluation included:

- To understand what anomalies could be detected,
- To understand whether and how proactive detection could be achieved,
- To evaluate the performance of the intelligent agent in terms of the number of anomalies detected, the average number of false alarms generated, the time of detection before existing alarm generation systems reported problems.

4.2.3 Data Input/Output Description

See [IA1]-[IA5] for descriptions.

4.2.4 Summary of Results

Two types of network anomalies were detected by agents: protocol implementation errors, and file server problems (due to excessive user requests). The ability of agents to detect anomalies **proactively** were tested from two aspects:

- If the agent can detect an anomalies earlier than the existing network scheme (through log messages or trouble tickets generated by end-users) on reporting problems, we say the agent can perform the detection proactively; and
- If the agent can detect a transient build up leading to a network problem such as file server problem due to excessive user request, we also claim that the agent can perform the detection proactively.

4.2.4.1 Case Studies of Typical Faults

Case Study (1): Protocol Implementation Errors

One of the anomalies detected on the enterprise network was a super server (inetd) protocol error. The existence of the fault was confirmed by syslog messages and trouble tickets. The syslog messages reported the inetd error. In

addition to the inetd error other faulty daemon process messages were also reported during this time. Presumably these faulty daemon messages were related to the super server protocol error. The trouble tickets also reported problems at the time of the super server protocol error. These problems were the inability to connect to the web server, send mail, print on the network printer and also difficulty in logging onto the network. The super server protocol problem is of considerable interest since it affected the overall performance of the network for an extended period of time. The detection scheme performed well on this type of error.

What is interesting about this case is the proactive detection capability of the agents compared with the existing trouble reporting systems. The detection by the intelligent agent was achieved (at night) about half an hour before the fault was reported through syslogs. The alarms generated by the intelligent agent were also persistent. Syslogs, on the other hand, could only report the problems in a passive fashion. As the protocol error was manifested at night when only a few users were accessing the network, syslogs were not able to report the problem as early as the agent was. The existing trouble-ticketing scheme only responds to the fault situation when users experience a network problem, and are therefore passive also. In fact, even if the agent started to generate alarms at night (and then syslog messages reported the problem), the first trouble ticket generated by a user did not come in until 11 O'clock in the morning. This example thus demonstrates how the agents can "proactively" detect a network problem compared with the existing trouble reporting systems.

Case Study (2): File Server Failures

In this case study we describe a fault scenario corresponding to a file server failure on subnet 2 of the campus network (see [IA1] for the network topology). 12 machines on subnet 2 and 24 machines outside subnet 2 reported the problem via syslog messages. The duration of the fault was from 11:10am to 11:17am (7 mins) as determined by the syslog messages. The cause of the fault was confirmed to be excessive number of ftp requests to the specific file server. As there was a transient time due to excessive user requests before the serve was done, the signature of the problem itself could be detected through traffic-related MIB variables. In fact, the fault was detected 21 minutes before the crash occurred. The mean time between false alarms in this case was found to be 1032 mins (approximately 17 hours). The persistence in the abnormal behavior of the router is also captured by the indicator. More details on this anomaly can be found in [IA1]-[IA5].

4.3 Discussion

Thus far the agent has been successful in identifying two different types of faults, file server failures and a protocol implementation error. The agent detected/predicted 9 file server failures on the campus network and 1 protocol

implementation error on the enterprise network. In both these cases the effects of the anomalies were observed in the chosen traffic related MIB variables. Also the changes associated with these anomalies events occurred in a correlated fashion, thus resulting in their detection by the agent. For the anomalies detected, the average time the first alarm generated by the agent was about half an hour before the existing system reported the problem. The average duration before false alarms was several hours. A more complete set of the anomalies and the performance of the agent in each case are discussed in the Appendices of [IA5].

4.4 Conclusions and Recommendations

Intelligent agents have been developed to generate alarms from measurements on MIB variables at routers and interfaces. The algorithm implementing the intelligent agents was based on statistical signal processing. The anomalies detected by intelligent agents included protocol implementation errors and file server problems due to excessive user requests. The intelligent agents were able to perform proactive detection by generating alarms much earlier than the existing trouble reporting systems (syslogs and user reports).

Our future work includes how to perform better detection and reduce false alarms using measurements from bursty variables (which could be a major source of false alarms), and how to better characterize a wider class of anomalies.

5. Visualization

The visualization tool provides a graphical display of the IA view of the health of the routers and interfaces being monitored. Since the IA is really a “change detector”, the visualization tool really displays structural changes in traffic patterns which may indicate a potential trouble spot. The status of the router and interfaces are represented as a gradient of colors. This gradient ranges from green, which indicates normal operation, through yellow and eventually red, which indicates very significant structural changes in the traffic pattern. This provides a user with a quick view of which areas on the network problems may be forming. For multiple routers, the physical connections between interfaces along with the respective health of each interface are represented.

5.1 Methods and Procedures

Once the visualization tool is launched, it periodically polls a lock file for information passed to it by the Intelligent Agent module. The network information is then read by the visualization tool and displayed in two windows. One is the functional window that has time stamp information and a series of control buttons. The other is a simple window that just maps the second network. A toggle button allows the user to switch network views between the primary and secondary windows. From the primary window, the user may also record

network information as it is being displayed. This recorded information can be accessed by the playback module, which allows the user to view the information anywhere in the file using a sliding menu bar.

The code is a series of script files written in Matlab that allows the visualization tool to be platform independent. In addition, it allows for quick proof of concept prototyping of new features that can eventually be added to a standalone executable.

5.1.1 Nature/Procedure of Data Processed and Algorithms Used

The data being processed consists can be found in four data files:

1. `iav_share.dat`: This file provides a share point of information between the Intelligent Agent and the visualization tools. Within this file there is data regarding the router numbers, interface numbers, status values and time stamps of all the router and interfaces to be displayed. This file gets updated periodically by the Intelligent Agent as accumulated information is ready to be displayed.
2. `iav_poll.dat`: In order for information to be properly accessed and shared using the `iav_share.dat` file there needs to be mutex lock. (Note: Mutex refers to mutual exclusion and is typically a flag that gets set and cleared by processes that share the same resource, thus, helping to avoid race conditions). The `iav_poll.dat` file provides that functionality. When the data to be displayed has been completely written by the Intelligent Agent, it sets a flag in the file to a locked state. Likewise, the visualization tool pools that lock and once it is set, it begins to read the `iav_share.dat` file. And when it is done, it will unlock the file by resetting the flag.
3. `iav_init.dat`: Data that both the Intelligent Agent and the visualization tool need in order to properly initialize is found in the file called `iav_init.dat`. This file has information about the router IP addresses, the router number that represents the router, the number of interfaces on the given router, which network the router belongs to that will be displayed, and the name of the DOORs configuration file used by the Intelligent Agent to obtain polling data.
4. `iav_ipinit.dat`: This file contains the information concerning the mapping of the IP address and the particular router and interface numbers associated with that IP address.

5.1.2 Software Flow Description

The interoperability of process flow diagram of the different Matlab modules used in the visualization tool can be referenced in the Software/Usage Document under the Visualization section of the introduction. The basic initialization and control of the software is managed by the main routine called `darpa1.m`. This launches the visualization tool by reading the initialization data files and starting the controls on the primary display window. It also launches the secondary

window for the other network to be displayed. The main loop then begins by polling the iav_poll.dat file to see if data is ready to be displayed. If data is there, it will construct the appropriate network display features ranging from assembling the router/interface shapes, mapping appropriate colors, determining the physical link coordinates, assigning labels and timestamps, etc. At that point, the router will display the appropriate constructed network. The main loop also checks to see which network is selected for viewing in the main window.

The main loop also checks if any of the record features have been activated. If the user selects to record the data as it is being displayed, they can do so by filling in the name of the binary data file they wish to create and pressing the record button. The display routine controlling the primary display window will then begin the process of setting up a binary file and appending to the end of the file data to be recorded. When the user wishes to display a previously recorded session, they can press the playback button in which case the main loop will launch a playback window. This window exists independently of the main windows and can therefore display the recorded data while new data is being collected and recorded into another file. This versatility allows the user to perform several tasks while still monitoring the health of the overall network. Pressing the exit button, which frees up any resources associated with that window, will close the playback window.

5.1.3 Development Environment Description

The development environment consists of a Sun Sparc Ultra server running the latest V5.3 r11 release of Matlab and version 5.6 of Sun Solaris operating system. Prototyping of the code was done using a series of dummy network data files complying with the visualization section of the API description found in the Software/Usage Document.

5.2 Experiments and Results

5.2.1 Experimental Environment Description

The visualization tool was tested using Matlab running on a Sun Sparc Ultra Workstation.

5.2.2 Methods/Parameters Evaluated

All of the functions that the visualization tool supports were tested.

5.2.3 Data Input/Output Description

See to section 5.1.1

5.2.4 Summary of Results

The basic integrity of the code was verified by running it for several hours with various data files.

5.3 Discussion

The intent of the visualization tool is to provide the user with high level visual information about the status of the network devices. This includes the router and the interfaces on the router. In addition, the visualization tool can be used to display connection information between interfaces on the network. As the complexity of the network being displayed increases, being able to quickly isolate potential network problems is very important. The record feature also is important in helping the user monitor network conditions over time and then quickly reviewing what occurred and when. This allows a user to observe the status trends over time, which can further help isolate problems.

5.4 Conclusions and Recommendations

The visualization tool is a prototype written for Matlab. This means the as the complexity of the display area increases, there will be performance issues that need to be addressed. Once all of the features have been solidified, it would be useful to write a standalone executable written in a platform independent language like Java. Added features to the display could include incorporation of more complex network topologies with scaling features. Also, having access through the visualization tool to the raw data, like Syslog messages, that were used by the Intelligent Agent in devising the error weights would be useful when investigating alarm conditions.

6. Network Control

Adaptive routing is a proactive approach to problem avoidance by routing packets away from “problem areas”. In our research, problem areas are indicated by the IA status values; these values are available at routers for this purpose. OSPF is a widely used method for building routing tables based on standard link metrics. We developed new link state metrics based on the IA outputs to be used by OSPF; this enables traffic to be automatically routed around impending problems before they become serious.

Traffic management controls the amount of traffic in a “problem area” by cutting back the traffic, rather than re-routing it. Two methods of traffic management were examined. Adaptive flow control for TCP/IP sessions is achieved by using the IA output to determine a “fair share” maximum window size for traffic through the router the IA is monitoring; that size will replace larger window sizes in TCP ACK packets passing through the router. Finally, admission control to a network region is limited by ingress routers in the presence of congestion: IA output indicating a congestion threshold has been exceeded results in setting the single bit Explicit Congestion Notification (ECN) on IP headers. This information is then sent to the ingress routers, where admission to the network is limited in response to the congestion.

Adaptive routing and Traffic Management are described separately in this section.

6.1 Adaptive Routing

In terms of the applicable scope, there are two broad categories of Internet routing protocols: inter-domain routing protocols that facilitate routing among different autonomous domains, and intra-domain, or interior routing protocols that are used to determine routing within an autonomous domain.

Within an autonomous routing domain in an IP network, for each class of services, a link in the network is assigned a non-negative cost. Routing tables are generally computed independently by each router using shortest path algorithms. The length of the path is the sum of the costs of the links consisting in the path. Therefore, changing the link costs will effectively change the routing within an autonomous routing domain.

Some interior routing protocols, such as Cisco's proprietary IGRP and EIGRP assign dynamic link costs that are variable based on the condition of the link. Open routing protocols, such as RIP, RIP2 and OSPF and IS-IS, do not define link cost as part of the protocols. Although link costs used in these protocols are generally configured statically, either manually or using default values, the protocols do not preclude dynamically changing the link costs when the need arises. In link state routing protocols such as OSPF and IS-IS, link costs can take on any positive integer values within a large range, thus allow the possibility of having many different routing choices by virtue of different link cost settings of the same physically (or logical) network topology. It is this possibility that we exploit as a technique of controlling network *routing without introducing architectural changes to the current routing framework*.

6.1.1 Methods and Procedures

6.1.1.1 Nature/Procedure of Data Processed and Algorithms Used

In the context of the Proactive Network Management architecture, the routing control component incorporates the probability of network anomaly from the Intelligent Agent in real time. It then uses this probability as an indicator to activate an algorithm to search for the optimal setting of link costs based upon the new network dynamics. This is illustrated in Figure 6.1.1.1.

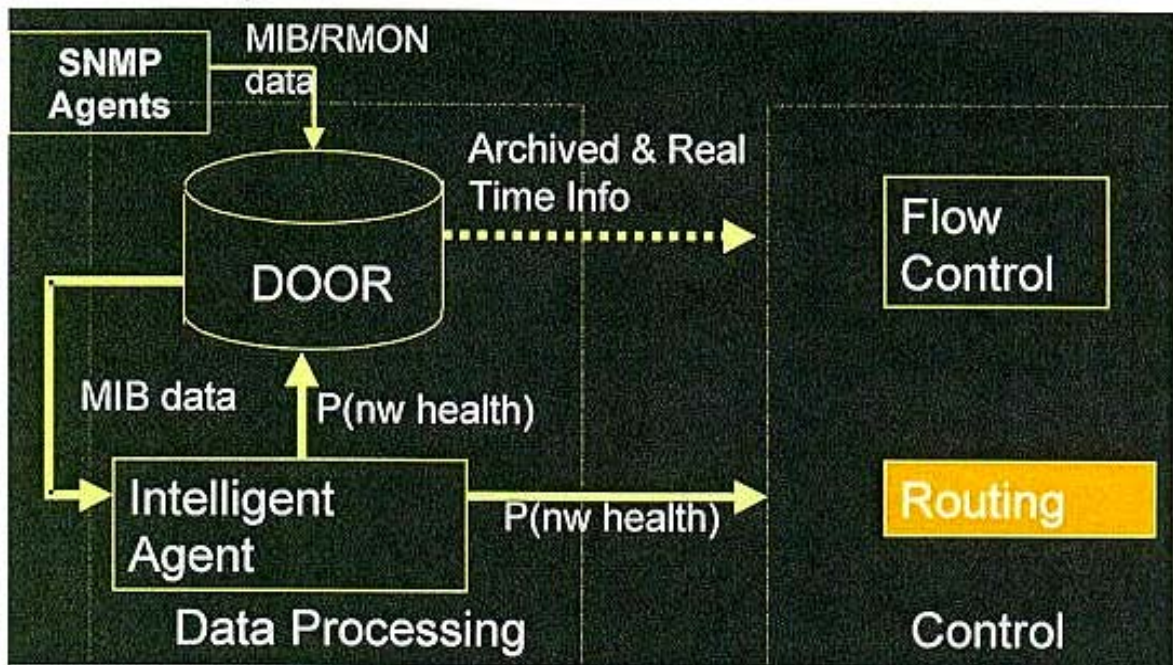


Figure 6.1.1.1 Routing Control in Proactive Network Management Framework

Both distributed and centralized versions of this adaptive network metric technique were designed, implemented and evaluated.

In the distributed algorithm, each router monitors its out-going links and determines, based on local information about the utilization of its own links, the corresponding link costs it will use and advertise to other routers. The mapping from the link utilization to link cost is a non-decreasing function to reflect the idea that a more heavily loaded link is less desirable compared to a less loaded one. To dampen routing oscillation and ensure convergence, a number of techniques are employed. These include exponentially averaging and thresholding the utilization, quantizing and, thresholding and upper-bounding the link costs, and regulating link cost change periods. The propagation of the new link costs and route calculation is done by any suitable routing protocols such as OSPF. We have implemented and evaluated in simulation HNcost, which is an example of such a protocol. The design and experimental results of HNcost are discussed in the next subsection.

The centralized algorithm fits in a network management architecture where there is a management station for the entire routing domain if the domain is small enough and has a flat topology or one management station for each routing area

if the network is hierarchical. Remote Network Monitoring (RMON) devices collect (samples of) source-to-destination traffic information and, either directly or through DOORS, report the information to the management stations. Each management station performs a search of the optimal setting of link costs for all the links in its domain when the need arises. If any of the link costs are changed, the management station installs the new link costs by setting the corresponding MIB variables using SNMP. We have developed and implemented Metricman as an example of such protocol, and have evaluated it in simulation. The design and experimental results of Metricman are discussed in the next subsection.

6.1.1.2 Software Flow Description

Network control techniques were studied in simulation experiments using the UCB/LBNL/VINT Network Simulator, version 2, ns-2¹.

HNcost is the name of our ns-2 implementation of the distributed adaptive metric algorithm.

The basic operation of HNcost is as follows:

1. When activated, HNcost in a node monitors the utilization and queue length of its out-going links and keeps the exponential averages of these quantities.
2. HNcost periodically checks the average utilization and queue length against thresholds to decide if calculation of new link costs is necessary. If so, go to (3). If not, go back to (2).
3. Check if minimum link cost change interval threshold is crossed. If not, go back to (2). If so, go to (4).
4. Calculates the target new link costs based on the configured mapping function. Regulate the target new link costs by a set of rules, such as maximum cost change, minimum cost change, change step size, to obtain the final new link cost.
5. Install the new link cost and notify the routing mechanism of the changes.

Metricman is the name of our ns-2 implementation of the centralized metric algorithm.

The basic operation of Metricman is illustrated in Figure 6.1.1.2 and summarized as followed below:

¹ <http://www-mash.cs.berkeley.edu/ns/>

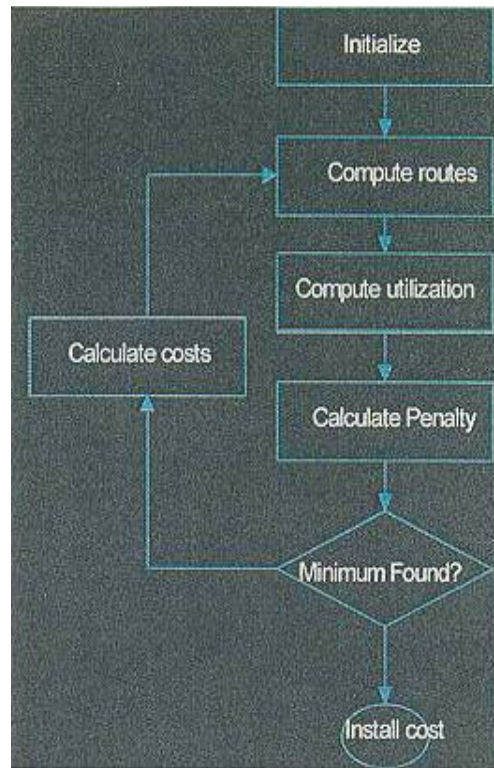


Figure 6.1.1.2 Algorithm of Metricman

1. Acquire the current topology. Gather source to destination traffic rates for all sources and destinations.
2. Then, when activated, copy the current setting of link costs to the “new” setting of link cost.
3. Use the new setting of link cost to compute the shortest path routes used for all source and destination pairs.
4. For each of the source destination pair, cast the corresponding traffic volume to each link along the route. In case of multiple routes with equal routes, traffic is split among the routes.
5. Sum up the traffics caused by all source and destination pairs to get the utilization of each link.
6. Compute the value of objective function of utilization and link cost.
7. If a minimum is found, go to 8. Other wise, go to 7.
8. For each link, map the utilization of each link into a new link cost. Then go to 3.
9. Install the new setting of link cost and call the ns-2 dynamic interface call back function to notify the changes.

The existing dynamic routing protocol in the network will then calculate the routing table and propagate the changes of link costs throughout the network. We developed and implemented in ns-2 such a dynamic routing protocol called rtProtoLS.

In terms of the actions it performs, rtProtoLS is designed to be a much-simplified OSPF-like protocol, and therefore, like OSPF:

- Each node sends out link state advertisement (LSA) to its peers when its link-state changes, or every 30 minutes on average by default.
- Each peer acknowledges the LSA, relays the new ones to its own peers except the ones that relay the same LSA to it before.
- All nodes' LSA are flooded through the network initially to form the topology database in all the nodes, which apply the Shortest Path First algorithm to calculate the next hop(s) to all destinations in the networks.
- In addition, when a link comes back up, the nodes at both ends of the link will exchange their topology database to see if there's anything new there. If so, it will regenerate the appropriate LSA and send it out to other neighbors.
- All unacknowledged LSA and Topology messages will be resent after a timeout. This timer will be canceled if the link to the peer goes down.

6.1.1.3 Development Environment Description

The simulation components were developed by extending ns-2 version 2.1b4. The components were developed on Solaris 2.6 running on Sun Ultra 10 computers, and on Linux RedHat 5.0 running on Intel Pentium processors. The program editors were emacs and xemacs. The compilers were g++ 2.8.0 and g++ 2.8.1. In addition, nam, perl5, tcl/tk 8.0 were used as development and testing tools.

6.1.2 Experiments and Results

6.1.2.1 Network Control Simulation Experiment Environment

Simulation experiments were run on non-hierarchical topologies for both HNCost and Metricman. The topologies were either randomly generated or taken from real topologies, such as the old NSFNET backbone and the ARPANET topologies. These topologies typically have fifteen to fifty nodes, with degrees of connectivity at about two.

The simulated traffic types were mixtures of random traffic generators with UDP transport, and simulation model of application protocols, such as Telnet, HTTP, FTP, using TCP as their transport mechanism.

The link state routing protocol used in the simulation to propagate and compute routes is rtProtoLS resembles OSPF in a flat topology with point-to-point links. It

was developed specifically to support investigation of adaptive metric for this project, but was also contributed to the ns-2 user community as the only link state routing protocol in ns-2.

6.1.2.2 Methods and Parameters Evaluated

Simulation experiments were conducted to identify the major factors that determine the effectiveness of the proposed network control mechanism.

The following factors were considered:

- Characteristics of the network topology. (e.g., size, speed, connectedness, symmetricness).
- End-to-end traffic pattern. (e.g., local vs. long distance pairs).
- The presence of TCP flow control.

In addition, different settings of the configuration parameters of the Metricman and HNCost are evaluated to gain insight to the network's reaction to the control mechanisms.

6.1.2.3 Performance Measurements in Experiments

For each simulation run, the following aggregate measurements were taken every simulation sample interval.

- Percentage of the packets dropped by the network.
- End-to-end delay averaged over the UDP packets.
- TCP retransmission rate TCP round trip time estimates.

6.1.2.4 Summary of Results

Case Study

We first discuss the results from a case study to illustrate the key observations, followed by general results from more simulation experiments.

The topology used in case study is modified from the old NSFNET backbone, with fourteen nodes and nineteen links, as shown in Figure 6.1.2.4.1. Link speeds were set at 56,000 bits per second. Propagation delays were set to approximate the actual propagation delay in the real topology. Random traffic was generated using a Pareto traffic model with UDP transport to simulate aggregate traffic. Average rates of traffic between any two nodes were set to be the same and at a level such that the average link utilization was about 40%. In addition, long distance Telnet sessions were put between selected nodes as probes that collected TCP performance statistics. The simulation was run for 2000 seconds, with Metricman activated at the 1000th second. The link state routing protocol for ns-2, rtProtoLS, was used to propagate and compute routes at simulation startup and after Metricman recomputed new link costs.

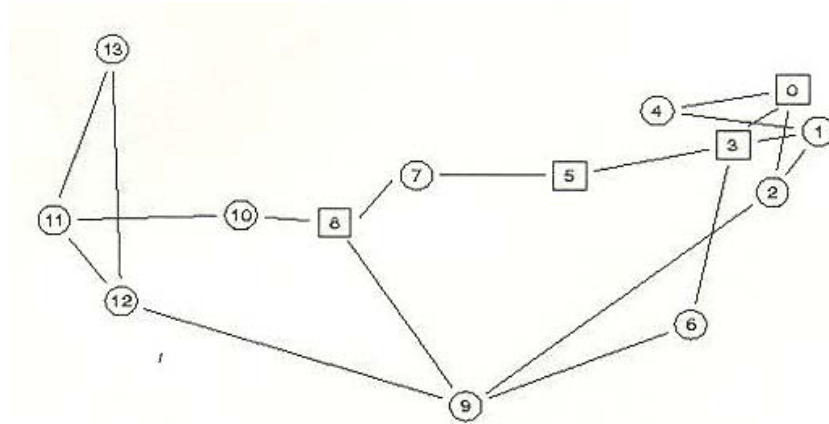


Figure 6.1.2.4.1 NSFNET backbone topology

Figure 6.1.2.4.2 shows the time series of the link utilizations. The statistics were collected every 15 seconds. Recall that new link costs were computed by Metricman at the 1000th second, the maximum link utilization dropped down from around 110% to around 90% shortly afterwards.

The average link utilization increased slightly for two reasons. First, fewer packets were dropped (which we will show in the next figure). This means that more packets stayed in the network. Second, some of the packets were routed away from the least hop count path, and thus traversed more hops than minimum. The standard deviation of the link utilization did not show significant change after Metricman was activated. Thus, we see that Metricman balanced the network traffic by reducing the load on the most heavily loaded link. It also did this without inducing significant extra traffic on the network.

As a result of such traffic balancing, packet loss were greatly reduced. Before activation of Metricman, packet loss in the most loaded link was at around 10%. Shortly after activation of Metricman, packet loss was eliminated for the rest of the duration of the simulation, as shown in Figure 6.1.2.4.3. This is because the traffic was set to be relatively stationary to better illustrate the long-term trend in the network. The temporary outburst of packets is queued in the link buffers with sizes of 50 packets and thus did not cause packet loss after the 1000th second.

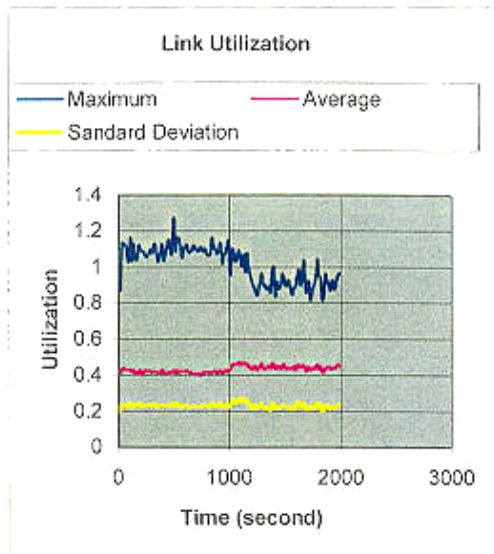


Figure 6.1.2.4.2 Link Utilization, before and after activation of Metricman

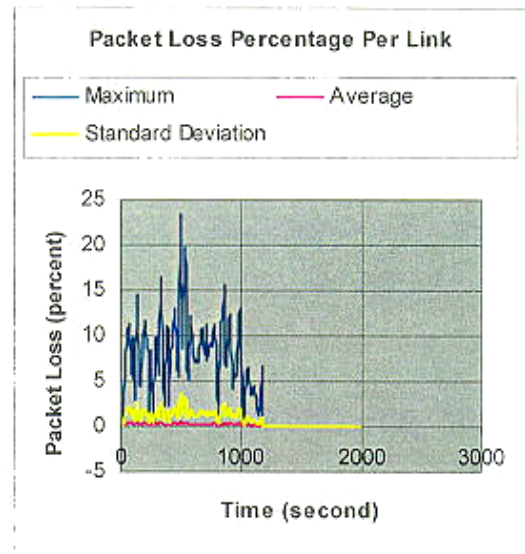


Figure 6.1.2.4.3 Packet Loss, before and after activation of Metricman

Reduced link utilization also translates into reduced queuing delay and therefore reduced end-to-end delay. This is demonstrated by the reduction of average UDP packet delay, shown in Figure 6.1.2.4.4, as well as the reduction of estimated Round Trip Time (RTT), shown in Figure 6.1.2.4.5, collected by TCP connections set up in the simulation as probes.

A number of observations can be drawn from the above case study.

1. The overall network performance, in terms of packet drops and delays, is largely determined by the most loaded congested link or links.
2. A small portion of packets traversing a few extra hops is more desirable than over-utilization of a few links in the network.
3. When we re-route some packets away from the most congested links, overall network performance is significantly improved in terms of packet drops and end-to-end delays.

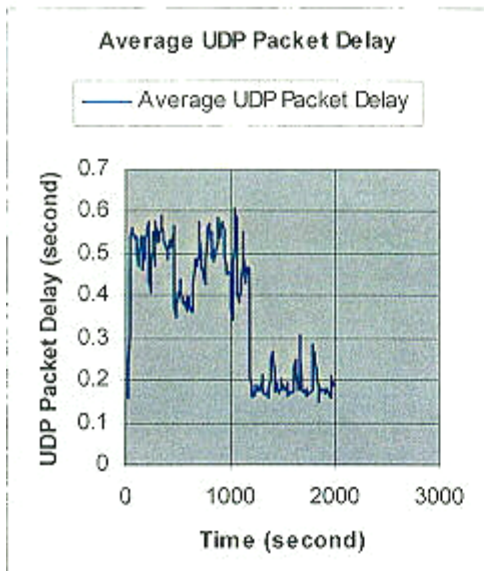


Figure 6.1.2.4.4 Average UDP delay, before and after activation of Metricman

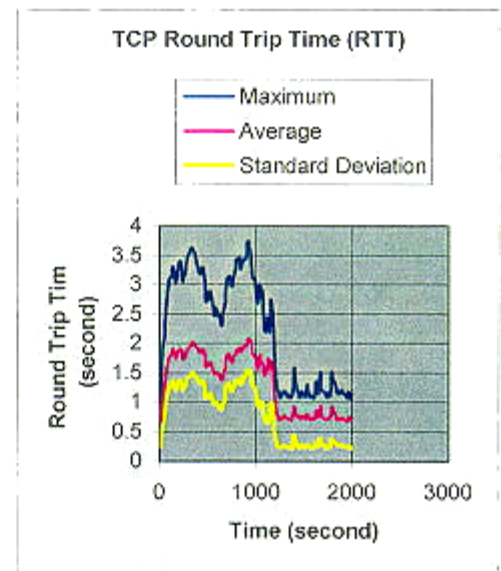


Figure 6.1.2.4.5 TCP round trip time, before and after activation of Metricman

Route Change on TCP retransmission.

To see the side effect of route changes on TCP connections, we look at the time series of retransmission rate of Telnet/TCP connections from the same case study in Figure 6.1.2.4.6.

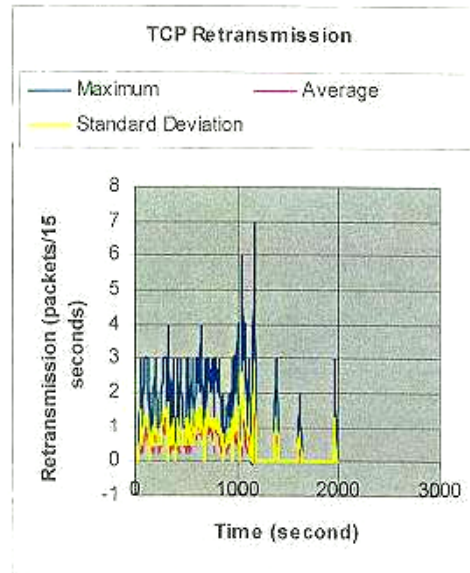


Figure 6.1.2.4.6 TCP retransmission

From the graph, we see that the TCP retransmission rate actually increased for the short time period immediately following the activation of Metricman. We theorize the explanation as follows. Due to changes that occurred throughout the network shortly after the 1000th second, some of the packets that had been queued up in the buffers now traversed extra hops to get to their destinations and therefore caused out-of-order arrivals. This triggered retransmission in classic TCP without Selective Acknowledgment (SACK). In the long run, however, the retransmission is drastically reduced to sporadic occurrences due to time-outs of the delayed packets.

Comparing Metricman and HNcost

Simulations with similar settings were run to evaluate HNcost, the distributed computation of adaptive link costs. Figure 6.1.2.4.7 and Figure 6.1.2.4.8 show the time series of total packet losses and average end-to-end UDP delays, respectively, for three of the experiments. In the first experiment, the default link cost was used throughout the simulation for 1000th seconds. In the second, HNcost was active from the 500th second till the end. In the third experiment, Metricman was activated once at 500th second.

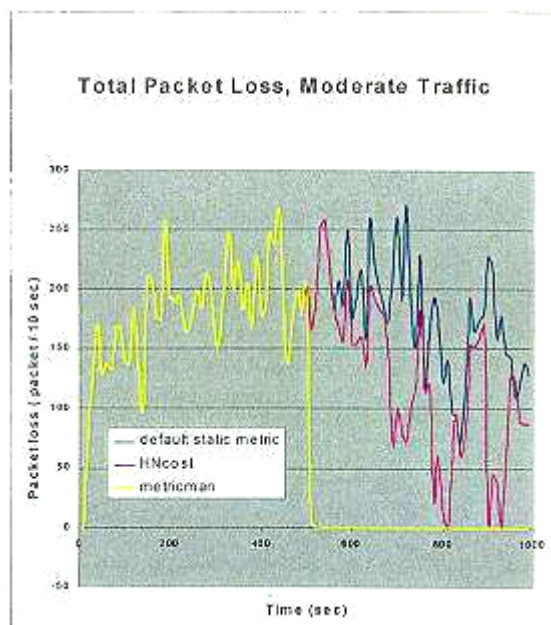


Figure 6.1.2.4.7
HNCost and Metricman
comparison - total packet loss

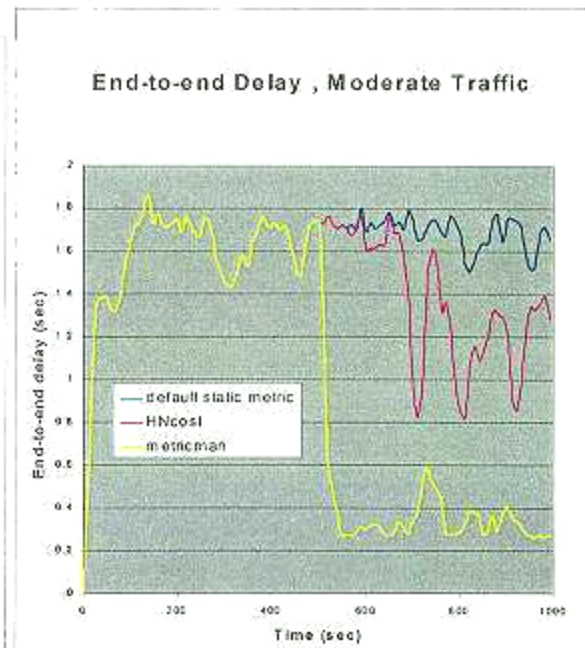


Figure 6.1.2.4.8 HNCost and
Metricman comparison -end-to-
end delay

We see from the figures that, with HNCost, while packet loss and delay were reduced to some extent compared to the case with default costs, the network did not reach a steady state even after a considerable time period (500th seconds). This contrasts with the consistent and stable improvement of network performance when Metricman was used.

The reason for the poor performance of HNCost becomes apparent when we look at the link cost evolution of one link, link 9-2 from node 9 to node 2, as shown in Figure 6.1.2.4.9. We see that the link cost slowly increased to 3570 before oscillating around 3000 and did not reach steady state.

The oscillation with HNCost can be explained as follows. The HNCost instance in node 9 re-evaluated the cost for the link 9-2 at every update interval (30 seconds), based on the weighted average of the link utilization over the past update intervals. HNCost instances in other nodes were doing the same thing for all other links in the network. When the link cost of a link was low, more traffic was routed to the link, which drove the link cost up. Once the link cost had become sufficiently high, some traffic was shed away from the link, which in turn drove the link cost back down. Without further stabilizing mechanism, the cost of the link oscillated between high and low with no guarantee to reach steady state.

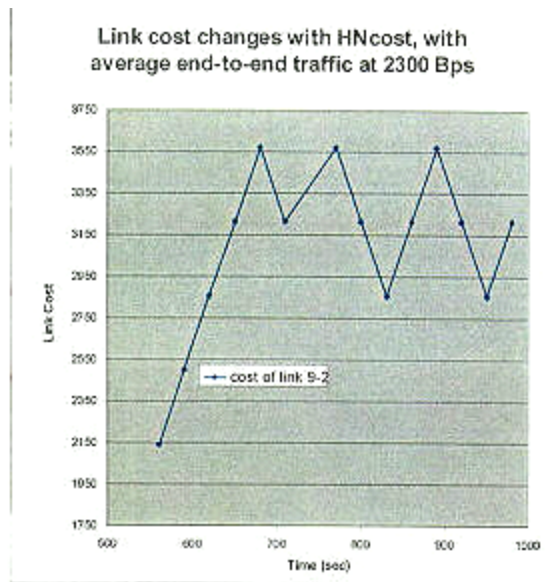


Figure 6.1.2.4.9 Link cost evolution with HNcost

Table 1. Link cost changes by Metricman for selected links.

Link Id	Old cost	New Cost
2-9	1750	3570
12-9	1750	3570
9-2	1750	3570
9-12	1750	3570

Table 6.1 Link cost changes by Metricman for selected links

The primary limitation of the distributed computation approach in HNcost is that each instance of HNcost has only local information and thus needs to wait for feedback from the network to determine the next step of action. Metricman, on the other hand, has the global information of traffic flows and performs the link cost iteration internally. Table 1, for instance, we see that Metricman changed the costs of the links between 2-9 and 9-12 from 1750 to be 3570 (other link costs not shown). It then installed the changed link cost in the network exactly once. This approach eliminates the possibility for link cost oscillation.

The stability issue of HNcost is being studied in a separate DARPA project². We now discuss other experimental results only for Metricman.

² "Network Management and Control Using On-line Collaborative Simulation".

To better understand the condition, under which routing control can be applied to improve network performance, and to confirm the observations we drew in the discussion above, more simulation experiments were conducted. Specifically, experiments were conducted to investigate the effectiveness of Metricman against:

- Different topologies,
- Different traffic scenarios,
- The presence of flow control.

Metricman in Different Topologies

Random topologies were generated using GT-ITM, the Internet Topology Generator by Ellen Zegura at the Georgia Institute of Technology³. We use two of the random topologies to illustrate our discussion. The first one, N22_L30, shown in Figure 6.1.2.5.0, has 22 nodes and 30 links with a link-to-node ratio of 1.36. The second one, N26_L39, shown in Figure 6.1.2.5.1 has 26 nodes and 39 links, with a link-to-node ratio of 1.5.

Figure 6.1.2.5.0 and Figure 6.1.2.5.1 show the time series of link utilization before and after the activation Metricman at the 1000th second, for topology N22_L30 and N26_L39, respectively. The simulation setups in both cases were the same as in the case study discussed above, except for the placement of Telnet/TCP connections and absolute end-to-end traffic levels. The resulting link utilization, in terms of the average, maximum and standard deviation were similar in both cases before the activation of Metricman.

In the case of N22_L30, activation of Metricman did not have noticeable effects on link Utilization, whereas in the case of N26_L39, the maximum of the link utilization decreased from around 110% to around 90%. Although the link-to-node ratios, 1.36 for N22_L30 and 1.5 for N26_L39, for the two topologies are comparable, a more careful look at N22_L30 reveals an undesirable characteristics: there are two groups of nodes, as highlighted in the figure, which are connected serially to form two long paths. This topology does not provide sufficient alternative routes to the most congested link under uniform end-to-end traffic level for all nodes. Compared to N22_L30, N26_L39 has a slightly higher link-to-node ratio and a more balanced layout of the links. In other words, N26_L39 is better connected than N22_L30 and thus leaves more choices for routing control.

³ <http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html>.

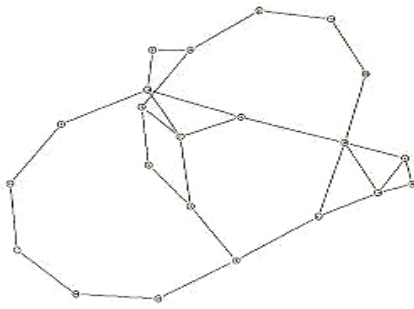


Figure 6.1.2.5.0 The N22_L30 topology. Link Utilization for N22_L30.

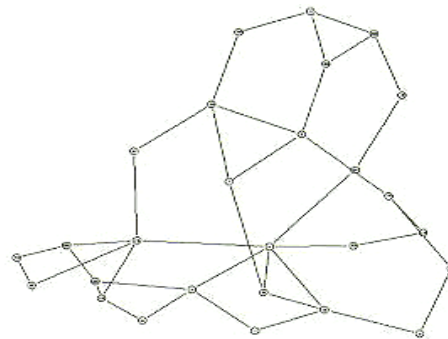


Figure 6.1.2.5.1 The N26_L39 topology. Link Utilization for N26_L39.

Metricman under Different Traffic levels

Simulation experiments were also conducted to evaluate the performance of Metricman under different traffic levels. In Figure 6.1.2.5.2, we show the time series of the maximum link utilization for three traffic levels. The topology is N26_L39 shown above. Other simulation settings were similar to the cases discussed above. The first series, *ave_util=55%*, represents an over-loaded network, with maximum link utilization at around 130% and average link utilization at 55% when using default link costs. The second one, *ave_util=45%*, represents a critically loaded network, with maximum link utilization just over 110% and average link utilization at 45% when using default costs. The third one, *ave_util=35%*, represents a heavily loaded network with maximum link utilization at around 90% and average link utilization at around 35% when using default link costs.

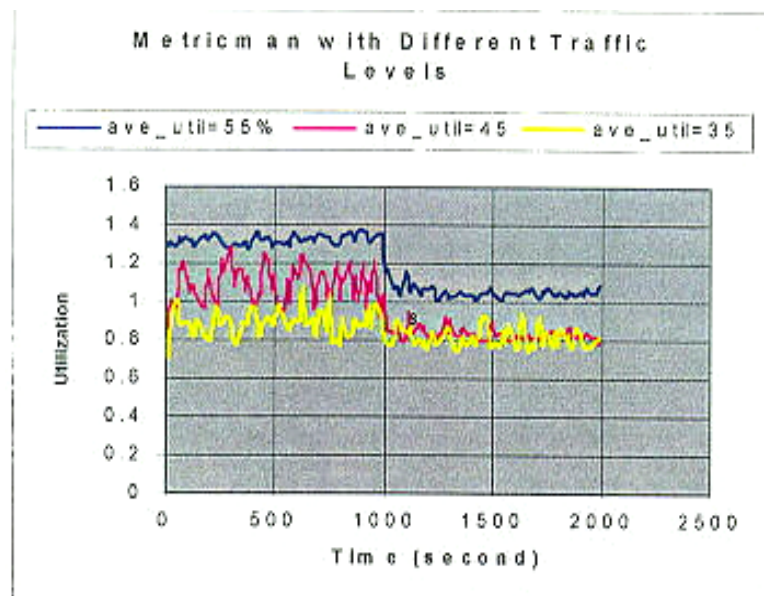


Figure 6.1.2.5.2 Metricman under different traffic

After activation of Metricman at the 1000th second, we see from Figure 6.1.2.5.2 that the maximum link utilizations in the over-loaded network dropped significantly from 130% to 105%. Even though the network is still over-loaded, the excessive packet arrival in the most congested link dropped from 30% to just over 5%. In the case of critically loaded network, maximum link utilization dropped from 110% to 85%, eliminating sustained excessive packet arrival. In the heavily loaded case, the maximum utilization also dropped from 90% to 80%. In other words, the network sees the most drastic improvement after the new routing when it was critically loaded. When the network is over loaded or just heavy load, Metricman can also reduce packet loss and delay significantly.

Metricman under Different Traffic Locality conditions

To study the effect of Metricman under different traffic locality conditions, we set up simulations to run under three types of end-to-end traffic conditions: *mostly local*, *mostly long distance* and *uniform*. In the mostly local traffic condition, the amount of traffic a node sends to its next-hop neighbor is about three times as much as the amount of traffic it sends to a neighbor 6 hops away. The opposite is true for mostly long distance traffic condition. In the uniform traffic condition, a node sends roughly equal amount of traffic to all other nodes.

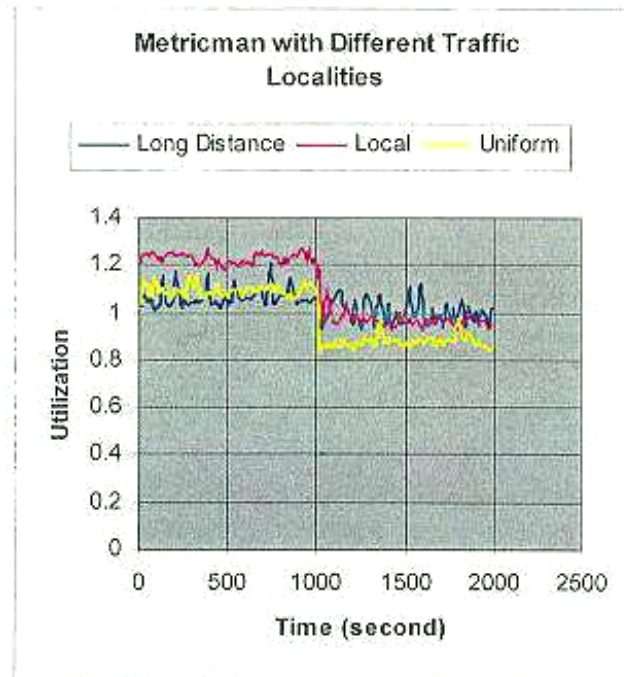


Figure 6.1.2.5.3 Metricman with different traffic localities

Figure 6.1.2.5.3 shows the time series of the maximum link utilizations for three simulation runs under mostly long distance, mostly local and uniform traffic conditions as described above, on the N26_L39 topology with similar setup as

other simulation cases. We see that Metricman significantly reduced the maximum link utilization under both uniform and mostly local traffic condition, but only reduced the maximum link utilization slightly (from about 105% to about 100%) in the case of mostly long distance traffic condition. Results from other simulations with different topology showed similar trend. The reason of such phenomenon is still under investigation.

Metricman and Long Lasting TCP Connections

We also ran simulations with traffic consisting entirely of long lasting TCP connections, instead of the UDP connections as in other examples discussed above. Figure 6.1.2.5.4 shows the time series of link utilization from one such simulation run in which the network was critically loaded. We see no significant changes in the maximum, average or the standard deviation of the link utilization after the activation of Metricman at the 1000th second. There were no significant changes in other performance indicators; they are therefore not shown here. Similar results were obtained from other simulation runs. This is because the long lasting TCP connections offer elastic traffic: traffic levels are mostly limited by the TCP transmission windows instead of by the amount of traffic generated by the pareto traffic model. In other words, if a link is not congested and therefore not dropping packets, TCP connections passing that link may keep sending more packets until the link starts dropping packets. Therefore, when the traffic models generated packets faster than TCP could send, the most congested links and their alternatives were all congested at similar levels. No alternative routing would have been able to change the situation significantly.

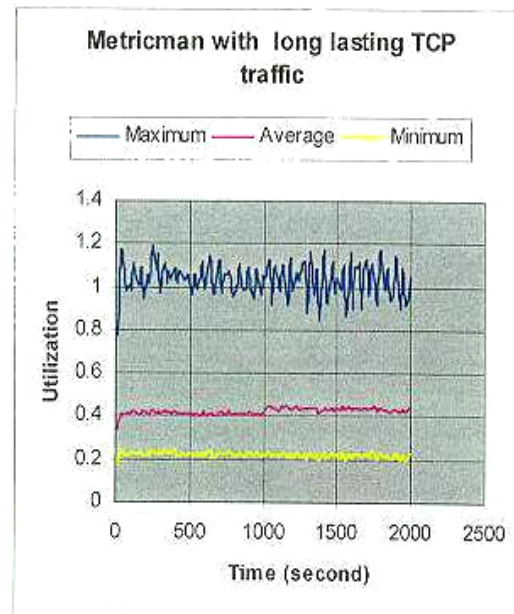


Figure 6.1.2.5.4 Metricman with long lasting TCP traffic

Metricman Parameter recommended range.

Other simulations were run to establish guidelines for the parameters in the Metricman. We found that Metricman was not very sensitive to small variations of the parameters as long as the parameters fall in the recommended ranges as listed in Table 2.

Table 2. Metricman parameter recommended range.

Name	Meaning	Recommended range
Max_hop	Maximum link cost in terms of the default link cost	2 – 4
Heavy_threshold	Threshold higher than which the load on the link is considered heavy	0.7 – 0.9
Step_size	Size of link cost change in terms of the default cost	0.2 – 0.5
Change_threshold	Minimum difference between the target cost and the current cost when a change is allowed	0.5 – 1
Light_threshold	Threshold under which the load is consider light	Less than 0.5

Table 6.2 Metricman parameter recommended range.

Summary of Results

To recapitulate the discussions above, we found that

1. Metricman, our coordinated adaptive link cost management scheme, combined with a link state routing protocol, can significantly improve network performance in terms of packet loss and end-to-end delay in well connected networks by balancing network loads, without introducing routing oscillation.
2. HNCost, the distributed dynamic link cost algorithm, can reduce packet loss and end-to-end delay to come extend, but it can take a long time to convergence and can lead to routing oscillation.
3. Route change can lead to temporary out-of-order packet arrival and causes more retransmission in TCP without selective acknowledgment shortly after the route change, but this short-term impact is compensated for by improved long-term performance of TCP in the form reduced round-trip time and retransmission rate.
4. Critically loaded networks (in which the utilization of the most congested links is just over 100% using default routes) see the most drastic performance improvement after activation of Metricman. In the case with well-connected networks that are either overly loaded (utilization of the most congested links much higher than 100%) or heavily loaded

- (utilization of the most congested links close to 100%), Metricman can also significantly improve the overall network performance.
5. We observed that networks with mostly long distance traffic see less performance improvement after activation of Metricman. The reason for this phenomenon is currently under investigation.
 6. We observed that in networks with long lasting TCP connections only, the most congested links are loaded at a similar level due to that fact that the TCP traffic levels are elastic and regulated by packet loss levels. Metricman was not effective in these special cases because there is no better alternative routing.
 7. Performance of Metricman is not very sensitive to small variations of the operational parameters as long as the parameters fall in the recommended range.

6.1.3 Discussion

Future Work in Routing Control: A number issues we briefly touch upon in our discussion above warrant further study. These Include:

- Stability of distributed computation of adaptive link costs (HNcost).
- Interaction between Metricman and TCP and other flow control mechanisms.
- Why networks with mostly long distance traffic seem to be less receptive to link cost management.

In addition, the Metricman algorithm can be studied and improved in the following directions:

- Experiment with different searching techniques to find the optimal link cost setting.
- Improve the scalability of Metricman in a large network by considering only a small subset of links.
- Analytically study the characteristics of network conditions that are best suitable for routing control.
- Study the effect of routing changes on stateful network protocols, such as RSVP and IP Multicast.
- Compare the technique of Metricman against other traffic engineering techniques such as IP tunneling.

Finally, routing control techniques in the inter-domain routing is another subject of future study.

6.2 Traffic Management

A new scheme called “TCP Rate Control” was developed. TCP Rate Control transparently augments end-to-end TCP performance by controlling the sending rate of a TCP source. The “rate” of a TCP source is determined by its window

size, the round trip time and the rate of acknowledgments (or ACKs). TCP rate control controls these aspects by modifying the ACK number and receiver window fields in acknowledgments and by modulating the acknowledgment rate.

From a performance viewpoint a key benefit of TCP rate control is to avoid adverse performance effects due to packet losses such as reduced goodput due to retransmission and timeout delays, and unfairness or large spread in per-user goodputs. Further, TCP rate control can positively affect performance even if the bottleneck is non-local and even if the end-host TCP implementations are non-conforming. This is done in a way to be fully compatible with the Intelligent Agent.

6.2.1 Methods and Procedures

TCP congestion control is designed for network stability, robustness and opportunistic use of network buffer and bandwidth resources on an end-to-end per-connection basis. Using a robust technique to detect packet loss, TCP infers congestion and trades off per-user “goodput” for network stability. Specifically, TCP throughput is known to be a function, which is inversely proportional to the round trip time, the timeout delays and the square root of loss probability [NC1].

Given this, we can view the function of any buffer management algorithm managing TCP flows as assigning p_i and queuing delays (which affect RTT_i) to competing TCP flows in order to meet performance requirements such as utilization, queuing delays, spread of per-user goodputs, etc., where p_i is the probability of a packet loss for flow i and RTT_i is the round trip time for flow i . However, this assumes that the TCP receiver window is not a limiting factor, which is not necessarily the case. Therefore if the TCP receiver window were the primary limiting factor, we could design a buffer management algorithm in which TCP throughput would not depend primarily on loss rate p_i (or RTT_i) under controlled operating conditions.

TCP rate control is motivated by these observations. Based upon rich congestion-related information available at the bottleneck, it first calculates and allocates rates to competing TCP flows. It then enforces these allocations by controlling the ACK number and receiver window fields in the ACK headers, and by modulating the ACK release rate. From a control-theoretic sense, the potential advantage of this approach is to have a higher observability of the control system, which could then lead to greater controllability. Observe that the rate allocation can be done by leveraging any of the well-known rate-calculation algorithms studied for the ATM ABR service [NC2]. Such algorithms have already been shown to achieve strong control of metrics such as utilization, queuing delay and fairness of allocations. The novel part in TCP rate control is to ensure that these properties are preserved in the rate-enforcement process.

6.2.2 Experiments and Results

The key contributions of this work are summarized below. For more details see [NC3].

We first articulated the issues in the rate-to-window translation/enforcement process. TCP rate control is best deployed at WAN edges of enterprise or ISP networks, not in ISP network cores.

Then we studied the performance of TCP rate control in comparison to RED and ECN [NC4, NC5]. When we divided our performance measures into two groups: user metrics and provider metrics, we found that, in general, RED and ECN optimize provider metrics, while TCP rate control optimizes both user and provider metrics.

More specifically, for long file-transfers, TCP rate control can manage the spread of per-user goodputs (a user metric) without compromising on utilization, aggregate goodput and queuing delays (provider metrics). An interesting aspect is that this can be achieved to a large extent, even if the round trip times (RTTs) of the competing flows are different. We verified the same result for other parameter dimensions such as homogeneous RTTs, LAN vs. WAN RTTs, and for different numbers of competing flows.

For short-file transfers, TCP rate control and TCP-ECN manage the spread of per-transaction transfer time (a user metric) better when compared to RED. But TCP-rate control does so by slightly reducing the total number of transactions completed (a provider metric), whereas TCP-ECN trades off a larger queue depth (a provider metric) for the same benefit.

Another key benefit of TCP rate control is in managing misbehaving (non-conforming) TCP sources, which can be easily created by hacking operating systems like Linux. Such sources if unchecked can combine to steal bandwidth from conformant sources (denial-of-service attacks), or even lead to congestion collapse. Assuming that such sources respond to ACK header fields (receiver window and ACK number), and then TCP rate control can successfully control them and eliminate all their ill effects. RED and ECN are much less effective in this respect.

TCP rate control can be reasonably effective even if the bottleneck is non-local; i.e. it is implemented not at the bottleneck but appears elsewhere in the path of flows. This feature is key to using TCP rate control in concert with the Intelligent Agent. This is because the Intelligent Agent might detect problems at an interior node where rate-control is not implemented, and end-systems may have been hacked.

6.2.3 Discussion

TCP rate control is a new technique, which transparently augments TCP performance through indirect control of its rate (achieved by manipulating the ACK stream only). Performance-wise, it avoids the adverse performance effects due to packet losses. Further, it can provide protection against misbehaving (non-conformant) TCP sources and improve performance even if the bottlenecks are non-local. Though it is somewhat constrained in its deployment space compared to stainless algorithms like RED, it can be (and has been) quickly and effectively applied at network edges (of ISPs and enterprise networks). It can be integrated with the Intelligent Agent to incorporate the agent output in a similar manner as done for routing (see Section 6.1).

6.3 Conclusions and Recommendations

In summary, we found that adaptive link cost is effective in balancing traffic in lightly to moderately loaded, well-connected networks. Calculation of the optimal link cost in a distributed fashion, however, can lead to oscillation of routes and is therefore not suitable in current Internet environment. A centralized dynamic link cost management scheme, combined with Intelligent Agent input, can be effective as a proactive network management tool in correcting local congestion related anomalies.

7. Design of Experiments

Wide area anomaly detection is based on the use of network probes to characterize expected performance ranges, and the use of a smaller set of optimally selected test probes to monitor current performance. This component focuses on the appropriate means of network characterization and optimal probe selection.

Users are interested in the latency or delay and also in specifying a suitable route (or path; we will use path and route interchangeably) by identifying congestion bottlenecks, if any, in the network. In this research, we concentrate on monitoring the network to answer two questions in real-time, namely, what will be the delay in sending a message from a source to a destination, and, where are the “hot-spots” or anomalies in the network.

One way to answer the two questions is by continuously monitoring the network and making predictions about the end-to-end delay and the location of the “hot-spots”. This monitoring requires sending probes between all sources and destinations in the network, which could potentially result in heavy traffic by just using the probes. A crucial decision is to choose an optimal subset of probes that is both small enough to handle (small files to store and browse through

quickly for monitoring) and not contribute to the congestion as well as large enough to make accurate predictions.

In this program, we use “traceroute-like” probes that give round-trip delay information between the source and every node in the route traversed by the probe from the source to the destination. Given these “traceroute-like” probes, the network topology, and, the routes followed by packets between all sources and destinations, we planned to formulate and solve a graph theoretic problem to optimally select a subset of probes.

7.1 Methods and Procedures

This graph-theoretic problem that we call the “constrained coverage problem” is an important contribution to graph theory literature. We also develop a sub-optimal polynomial time algorithm to solve the problem. Using this algorithm we obtain a set of probes that can be used to continuously monitor the network to obtain information about the roundtrip delays between any two nodes and also to identify the “hot spots” in the network.

7.1.1 Nature/Procedure of Data Processed and Algorithms Used

The objective of this study is to select the optimal or sub-optimal subset of source and destination (S-D) pairs for continuous probing. The delay data obtained from the probes sent between the subset nodes will be used to calculate the delays between any S-D pair in the topology.

The probe type can be one of many alternatives (traceroute, ping, etc.) each of which provides different levels of details for the delay information. For example, ping provides round-trip end-to-end delay, but no information about delays arcs along its path from the source to the destination. On the other hand, traceroute provides round trip delays from the source node to all the nodes along its path. Therefore, in this study, we use traceroute-like intelligent agents as the probe type.

7.1.2 Software Flow Description

No software was developed for the DOE component under this program.

7.1.3 Development Environment Description

7.2 Experiments and Results

Preliminary experiments were on correlation of probes through “nearest neighbor” routers and adjacent paths through the Internet was begun, and data collected. Analysis of that data was not completed due to the cutback in this area.

7.3 Discussion

The work done on this program in probe selection is valuable, but theoretical in nature. To apply this work to networks, a better understanding of the correlation relationships among performance metrics (such as delay) of traffic in the Internet is critical. Although there has been a large body of mathematical and simulation models of the IP networks, none provided a basis for these correlation models. This determination meant that practical results could only be obtained with a considerable secondary effort into this type of modeling, which was beyond the budget available. In light of that, even the probe selection work was stopped on this contract.

PSU did continue the theoretical research under internal funding, and that work is documented in [DOE 1]

7.4 Conclusions and Recommendations

The work begun in this area has significant potential, and should be continued with higher levels of funding than were available. In particular, it is recommended that efforts be funded in two areas:

- Develop both theoretical and practical understanding of the correlation of network performance behavior.
- Extend the existing probe selection work a more practical domain by making use of these correlation models.

8. Proactive Connection Request Manager

The Proactive Connection Request Manager (PCRM) is a generic approach to using information provided by the DOE component to select preferred network entry points/modes and target hosts for services run across a large IP network. The preferred path is one which is expected to meet performance objectives, and may be selected partly on the basis of cost of service.

While the DOE approach was being developed, a simplistic version of the problem was analyzed: the selection of an egress HTTP proxy gateway from the Lucent Intranet to a target site across the public Internet. This is a non-trivial problem of interest given the large number of egress gateways from the international Lucent intranet, and the lack of any formal load balancing across them. In fact, Lucent personnel at times manually reconfigure their browsers to a different proxy rumored to be fast.

When the DOE component was cut, there was no longer a practical basis for data to drive the connection preferences, and further work on PCRM was stopped after the experiments described below.

8.1 Methods and Procedures

This section describes software and experiments run to evaluate the performance over different egress gateways to a set of external mirrored web

sites. From a single host, 15 gateways may be used; there are 5 mirrored web sites. A path consists of selection both an egress gateway and a destination web site. Which path provides the best performance?

8.1.1 Nature Procedure of Data Processed and Algorithms Used

The experimental data is based on the time it takes to send and then receive a response to an HTTP Get command. The sender consists of a host machine that simultaneously queries the proxy server and the destination server on the other side of the proxy server. The host starts a timer for each poll and stops the timer once the HTTP header information has been received. Several polls are issued at the same time to various proxy servers and destination servers. The results are then collected, parsed and plotted.

8.1.2 Software Flow Description

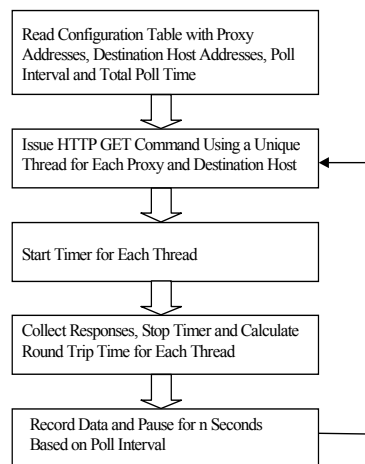


Figure 8.1. Software Flow Description

8.1.3 Development Environment Description

The test code was written using a PC running Microsoft Windows NT TM. The code was developed and compiled using Microsoft C/C ++ Developers Studio TM.

8.2 Experiments and Results

8.2.1 Experimental Environment Description

The test software ran on a Pentium based PC inside the proxy. The overall poll time was set for approximately one week at a time at a frequency of one poll cycle every 20 seconds. The poll time was altered to various lengths, as was the poll cycle. The number of proxy and destination servers was fixed (15 proxy servers, 5 destination servers). Thus, the same servers were always polled. The data was then collected and stored for further analysis.

8.2.2 Methods/Parameters Evaluated

The round trip delay time of each HTTP GET command and response was recorded throughout the day.

8.2.3 Data Input/Output Description

The query command consisted of a basic HTTP GET command issued to the proxy and destination server. The response consisted of the delay time it took for a response to occur from the proxy/destination servers, a time stamp of the query, the proxy/destination server names and any error message from the servers.

8.2.4 Summary of Results

The results indicate that throughout the day, certain proxies have less delay time in reaching the same destination servers. Heavily trafficked proxies show the greatest delay times. In addition, the delay time from the destination servers varies according to the time of day and time of the week. Trends indicate that early morning polls (between the time of 1 AM and 4 AM) and those that occur on the weekends receive the quickest response times from both the proxy and destination servers.

8.3 Discussion

These results demonstrated the great variability in performance between gateways and at different times of day. In terms of preferred path selection, even this simplistic problem becomes complex: with 15 egress gateways and 5 possible target servers, 75 combinations need to be tested and compared. The DOE component would allow tests on a smaller set of paths to be made, and a selection made based on projected performance of all paths.

8.4 Conclusions and Recommendations

The variation seen in even this simplistic example shows that a method to select preferred paths is valuable. Since evaluation of all possible paths is impractical, a method such as DOE to collect sample performance data, combined with correlation analysis tools should be pursued to enhance user observed performance in IP networks.

9. Project Conclusions and Recommendations

Valuable technologies for robust network management data collection, anomaly detection, and anomaly response have been developed. These technologies, particularly the detection and response technologies still need further analysis, and to be deployed and integrated in a large-scale test environment. More specifically, we recommend the following additional research areas:

- Develop an understanding of the types of anomalies detected by the agents. Currently, they effectively tell the user “something is abnormal” but do not indicate what -- or even if it is a real problem.
- Develop an understanding of the actions of multiple agents monitoring multiple network elements in a network region.
- Deploy the adaptive routing and adaptive flow control algorithms in a test network, and eventually in a real network. The current evaluations by simulation and analysis are a good start, but it is critical to see them work on real traffic -- and to see how real traffic reacts to their actions.
- Develop an understanding of the correlations of performance of neighboring network elements. This will be the gateway to a better understanding of network performance, and a valuable tool in many other network simulation and modeling research projects.
- Develop a DOE style model of network performance based on these correlation models.
- Develop a full PCRM system to provide **service** requesters the best possible performance through a complex network even when there are local problems in that network.

References

[DOE 1] Huseyin C. Ozmutlu, Russell Barton, Natarajan Gautam, William Hery, “Network Monitoring: Probe Subset Selection Using the Constrained Coverage Problem”

[DOOR 1] Alan Bivens, Li Gao, Mark F. Hulber and Boleslaw K. Szymanski, “Agent-Based Network Monitoring” Proc. ‘Agent based High Performance Computing’ Workshop at Autonomous Agents99 Conference, Seattle, Washington, USA, May 1-5, 1999, pp. 41-53.

[DOOR 2] Alan Bivens, Patrick H. Fry, Li Gao, Mark F. Hulber, Quifen Zhang and Boleslaw Szymanski, “Distributed Object-Oriented Repositories for Network Management”, Proc. International Conference on Software Engineering, IEEE Computer Science Press, Las Vegas, August, 1999, pp. CS169

[IA1] M. Thottan and C. Ji, “Adaptive Thresholding for Proactive Network Problem Detection”, Proceeding of IEEE International Workshop on Systems

Management, Newport, Rhode Island, April 1998, 108-116, also available from <http://neuron.ecse.rpi.edu/>.

[IA2] M. Thottan and C. Ji, "Statistical Detection of Enterprise Network Problems", Journal of Network and Systems Management, 1999, Vol. 7, No. 1, 27-45, also available from <http://neuron.ecse.rpi.edu/>.

[IA3] M. Thottan and C. Ji, "Proactive Anomaly Detection using Distributed Intelligent Agents", IEEE Network, Sept/Oct 1998.

[IA4] M. Thottan and C. Ji, to appear at Network Management Conference, 2000.

[IA5] M. Thottan, Ph.D. thesis, Rensselaer Polytechnic Institute, to appear.

[NC1] J. Padhye, V. Firoiu, D. Towsley, and Jim Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation", Proceedings of SIGCOMM'98, Vancouver, August 1998.

[NC2] S. Kalyanaraman, "Traffic Management for the Available Bit Rate (ABR) Service in Asynchronous Transfer Mode (ATM) Networks", Ph.D. Dissertation, Dept. of Computer and Information Sciences, the Ohio State University, August 1997.

[NC3] S. Karandikar, S. Kalyanaraman, P. Bagal, and B. Packer, "TCP Rate Control", Computer Communications Review, Jan. 2000, pp. 45-58, also available from <http://www.ecse.rpi.edu/Homepages/shivkuma/>.

[NC4] S. Floyd, and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, Vol. 1, No. 4, August 1993, pp. 397-413.

[NC5] K. K. Ramakrishnan, S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IPv6 and to TCP", IETF Internet Draft, November 1997, also available from <http://ds.internic.net/internet-drafts/draft-kksjf-ecn-00.txt>.

Symbols, Abbreviations and Acronyms

ABR - Available Bit Rate
ACK - Acknowledgement packet
AR - Auto-Regressive process
AS - Autonomous System
ATM - Asynchronous Transfer Mode
BR - Border Router
BRA - Border Router Adjunct

CIO - Chief Information Officer organization
CPU - Central Processing Unit
DOE - Department of Energy
ECN - Explicit Congestion Notification
CAIDA - Computer Aided Incident Data Analysis
CORBA - Common Object Request Broker Architecture
DOORS - Distributed Object Framework
DOORS - Distributed Object-Oriented Repository System
DOE - Design of Experiments
GT-ITM - Internet Topology Generator software
HNcost - a distributed, adaptive algorithm for computing link metrics for OSPF
HTTP - HyperText Transfer Protocol
IA - Intelligent Agents
IP - Internet Protocol
IP Multicast - A protocol for transmitting IP datagrams from one source to many destinations in a local or wide area network of hosts which run the TCP/IP suite of protocols
ISP - Internet Service Provider
LSA - Link State Advertisement
Metricman - a centralized metric algorithm for computing link metrics for OSPF
MIB - Management Information Base
MWS - Maximum window
OSPF - Open shortest path first
PMDB - Path metric database
PCRM - Proactive Connection Request Manager
PP - Probe Points
PSU - Penn State University
QoS - Quality of Service
RA - Router Adjuncts
RED - Random Early Detection
REDP - Random Early Detection Plus
RMON - Remote Monitor
RPI - Rensselaer Polytechnic Institute
RSVP - Reservation Protocol
RTT - Round Trip Times
S-D - Source and Destination
SNMP - Simple Network Management Protocol
SYSLOG - System Log
TCP - Transmission Control Protocol
TCP/IP - Transaction Control Protocol/Internet Protocol
TOD/DOW - Time of Day/Day of Week
UDP/IP - User Datagram Protocol/Internet Protocol
WAN - Wide Area Network